HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Kurt Junghanns

Analyse der OpenSource Mapserver GeoServer und MapServer für die Nutzung mit Agriport Mobile

2012

Fakultät
Mathematik/Naturwissenschaften/Informatik

BACHELORARBEIT

Analyse der OpenSource Mapserver GeoServer und MapServer für die Nutzung mit Agriport Mobile

Autor:
Kurt Junghanns

Studiengang:
Informatik

Seminargruppe:
IF09w1-B

Erstprüfer:
Prof. Dr. rer. biol. hum. Rudolf Stübner

Zweitprüfer:
M. Sc. Volkmar Herbst

Einreichung:
Mittweida, 5. Oktober 2012

Verteidigung/Bewertung:
Mittweida, 2012

Faculty
mathematics/natural sciences/computer science

BACHELOR THESIS

Analysis of the OpenSource Mapserver GeoServer and MapServer particularly for the use of Agriport Mobile

author:
Kurt Junghanns

course of studies:
Informatik

seminar group:
IF09w1-B

first examiner::
Prof. Dr. rer. biol. hum. Rudolf Stübner

second examiner::
M. Sc. Volkmar Herbst

submission:
Mittweida, 5. Oktober 2012

defence/evaluation:
Mittweida, 2012

Bibliografische Angaben

Junghanns, Kurt: Analyse der OpenSource Mapserver GeoServer und MapServer für die Nutzung mit Agriport Mobile, 85 Seiten, 24 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2012

Satz: LaTeX

Referat

Das Ziel dieser Arbeit ist es, anhand einer Analyse der OpenSource Mapserver GeoServer und MapServer, einen Zugang zu den Mapservern zu schaffen und eine Empfehlung für den Prototypen Agriport Mobile abzuleiten.
Der Autor schuf im Rahmen seines Praktikums den Prototypen Agriport Mobile, welcher durch die Hinführung und Empfehlung hinsichtlich der Kartenerzeugung verbessert werden soll.
Die vorliegende Arbeit ist bei der AgriCon GmbH entstanden. AgriCon ermöglicht Landwirten und Lohnunternehmern im Agrarbereich eine Erhöhung der Leistungsfähigkeit des Pflanzenbaus mit Hilfe von Precision Farming.

Bibliography

Junghanns, Kurt: Analysis of the OpenSource Mapserver GeoServer and MapServer particularly for the use of Agriport Mobile, 1 pages, Hochschule Mittweida (FH), Faculty mathematics/natural sciences/computer science

Bachelor Thesis, 2012

Abstract

The purpose of this paper is to create an introduction to the Mapservers and deduce a recommendation for the prototype of Agriport Mobile based on an analysis of the OpenSource Mapservers GeoServer and MapServer.
During his internship the author created the prototype Agriport Mobile, which should be improved by this analysis and recommendation relating to the generation of the maps.
This thesis was written for AgriCon GmbH. AgriCon enables agriculturalists and contractors at the agrarian sector to improve the effectiveness of crop farming via precision farming.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

AP	AgriPort
CGI	Command Gateway Interface
CQL	Common Query Language
DBMS	Datenbankmanagementsystem
EPSG	European Petroleum Survey Group Geodesy
FE	Filter Encoding
GIS	Geoinformationssystem
GML	Geography Markup Language
GS	GeoServer
GWC	GeoWebCache
IIS	Microsoft Internet Information Services
JDK	Java Development Kit
JNDI	Java Naming and Directory Interface
JPG	Joint Photographic Experts Group Norm
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KML	Keyhole Markup Language
MS	Mapserver
OSGeo	Open Source Geospatial Foundation
PDF	Portable Document Format
PNG	Portable Network Graphics
REST	Representational State Transfer
SWIG	Simplified Wrapper and Interface Generator
TC	Tile Cache
URL	Uniform Resource Locator
WMS-C	Web Map Service Tiling Client Recommendation
XML	Extensible Markup Language

1 Einleitung

1.1 Motivation

Precision Farming gewinnt in der modernen Landwirtschaft zunehmend an Bedeutung. Denn damit ist es möglich, die Einsatzkosten zu senken, den Ertrag zu erhöhen und gleichzeitig die Umwelt zu schonen.
Precision Farming bedeutet in diesem Zusammenhang eine individuelle Betrachtung und Bewirtschaftung einzelner Teile von Flurstücken, wodurch Unterschiede des Bodens und die variierende Ertragsfähigkeit innerhalb eines Flurstückes berücksichtigt werden.
Die Anzahl von 22.000 Landwirten und Fachleuten der Agrarwirtschaft, welche die DLG Fachtage im Juni 2012 besucht haben, zeigt, dass ein großes Interesse an modernen Verfahren und Technologien besteht[1]. Precision Farming vereint eine Reihe von modernen Verfahren und Technologien, weshalb die Aussage der Journalistin Röseberg nachvollziehbar ist:

> „Precision Farming ist nicht das Zauberwort der Produktionstechnik im Pflanzenbau, es ist die Zukunft.“[2]

Die Durchführung von Precision Farming erfordert die Handhabung aktueller Hard- und Software. Neben moderner Sensoren, Terminals und GPS-Technik sind leistungsfähige und kundenfreundliche Softwarelösungen notwendig, um bestmöglichste Ergebnisse erzielen zu können.
Es handelt sich um einen vollständig technischen Bereich, in welchem Informatik ein essentieller Bestandteil ist.

Das Mooresche Gesetz sagt aus, dass sich aller 12 bis 24 Monate die Komplexität interner Schaltkreise verdoppelt[3]. Entsprechend dieser Entwicklung steigt die Leistungsfähigkeit der Sensorig und der die Daten verarbeitenden Hardware. In Folge dessen erfolgt die Datenerfassung und teilweise auch die -verarbeitung immer mehr in Echtzeit. In Echtzeit heißt, dass automatisch und unmittelbar nach der Erfassung der Daten, diese verarbeitet werden. Dabei beträgt die Zeitdifferenz zwischen der Erfassung der Daten und der Verfügbarkeit der verarbeitenden Daten maximal 30 bis 60 Minuten.
Die Anforderungen an die Software steigen mit der Entwicklung der Hardware und den Möglichkeiten, die neue Sensoren, welche im Precision Farming eingesetzt werden, bieten. Es fallen pro Zeiteinheit mehr Daten an, wodurch ein hoher Durchsatz gefordert wird.
Die Software muss optimiert werden, um dem erhöhten Datenaufkommen durch die

[1] vgl. [AB12]
[2] [RÖ10]
[3] vgl. [ITW-MG]

Hardware gerecht zu werden.
Stabilität, Performance und Zuverlässigkeit wird in jedem seriösen Betrieb als Eigenschaft von Software gefordert. Jedoch sind diese 3 Eigenschaften im Besonderen für die Erfüllung der Forderung nach Echtzeitverarbeitung eine Grundvorraussetzung.
Um alle Forderungen zu erfüllen und gleichzeitig Kundenzufriedenheit und Interoperabilität zu gewährleisten, müssen die einzelnen Elemente der Softwareprodukte professionell analysiert, angepasst und gewartet werden.

Das Internet-Datenportal AgriPort der Firma AgriCon ist eine Plattform, die eine automatische Verarbeitung und Darstellung der Precision Farming - Daten ermöglicht.
Die erfassten Daten können in ihrer ursprünglichen sowie aufbereiteten Form eingesehen, analysiert dargestellt und ausgewertet werden.
AgriPort ist für den stationären Gebrauch an einem feststehenden Arbeitsrechner konzipiert.
Der Autor schuf im Rahmen seines Praxismodules einen Prototypen in Form einer Webanwendung, welcher einige Funktionalitäten des Datenportales für mobile Endgeräte, wie Smartphones und Tabletts, plattformunabhängig zur Verfügung stellt. Siehe dazu [JU12].
Ein Prototyp stellt keine fertige Software dar, wodurch Verbesserungen, Korrekturen oder einzelne Umstrukturierungen möglich sind.
Diese mobile Lösung ist bereits für Kunden nutzbar und sollte ansprechender gestalltet werden.
Um die Qualität der Software zu verbessern, soll der kritischste Qualitätsfaktor der Anwendung aufgegriffen werden. Dieser ist die Dauer und Qualität der Erzeugung sowie Anzeige der Karten.

Im AgriPort sowie in der in der mobilen Lösung, sind die flüssigen Kartendarstellungen dynamischer Inhalte Kernfunktionen, durch welche sich AgriCon von anderen Herstellern von Agrarsoftware absetzt. Somit bekommen diese Eigenschaften und die hier untersuchten technologischen Ansätze eine besondere Bedeutung.
Aus diesem Grund müssen die Karten nicht nur qualitativ hochwertig und exakt sein, sondern auch innerhalb von Momenten bereitstehen.
Zusätzlich handelt es sich um dynamisch erzeugte Karten, wodurch es notwendig ist das Softwareelement zu verbessern, welches die Daten aus der Datenbank liest und aus diesen entsprechend den Wünschen des Nutzers die Karten erzeugt.
Um sich im mobilen Bereich ebenso von anderen Herstellern abzusetzen, ist es erforderlich, die flüssigen Kartendarstellungen dynamischer Inhalte ebenso durch die mobile Lösung zu realisieren.

1.2 Zielsetzung

Das Softwareelement, welches sich zwischen der Datenbank und der Anwendung beim Client befindet, ist ein Internet Map Server.
Aus den genannten Gründen ist es notwendig, für den Einsatzzweck der mobilen Lösung eine Analyse von geeigneten Servern durchzuführen und daraus eine Empfehlung für die Nutzung abzuleiten.

In dieser Bachelorarbeit wird für diesen Zweck eine Analyse der Internet Map Server, kurz Mapserver genannt, mit anschließendem Vergleich durchgeführt. Die Analysepunkte sind dabei die Einstellungsmöglichkeiten für dynamische Karten und die Leistungsfähigkeit.
Weiterhin erfolgt die Analyse in Hinblick auf den Prototypen.
Es soll an Hand dieser Arbeit möglich sein, für den Prototypen einen geeigneten Mapserver zu konfigurieren und zu verwenden sowie wichtige Schlüsse für AgriPort zu ziehen.
Es findet eine Orientierung an einer prototypischen Implementierung eines Mapservers für AgriPort statt, jedoch soll dadurch keine zwingende Abhängigkeit entstehen.
Der Prototyp verwendet den Dienst WMS und die Datenquelle PostGIS, weshalb darauf ein besonderes Augenmerk in der Analyse gelegt wird.
Dabei werden außerdem Hinweise zu den Möglichkeiten von WMS gegeben, welche vom Prototypen momentan nicht genutzt werden.

Im Anschluss an die Einleitung werden die für das Verständnis notwendige Begriffe geklärt.
In den zwei darauf folgenden Unterkapiteln wird die Verwendung eines Mapservers aus Sicht des Datenportales AgriPort und anschließend aus Sicht des Prototypen dargestellt.
Vor Beginn des nächsten Kapitels wird die Testumgebung definiert und dessen Aufbau begründet.
Das dritte Kapitel widmet sich ausschließlich dem Mapserver GeoServer und dessen Merkmalen.
Kapitel vier analysiert hingegen den Mapserver MapServer, sodass im fünften Kapitel ein Vergleich vollzogen und anschließend ein Fazit gegeben werden kann.

2 Problemanalyse

2.1 Mapserver

Mapserver (MS)
Es existiert keine allgemeingültige Definition des Begriffes.

> „Ein MS ist ein Programm, welches der interaktiven, individuellen und unmittelbaren Erstellung und Visualisierung von geographischen Informationen in Form von Karten über das Internet dient.“[4]

Ein MS wird auf einem Server installiert und stellt seine Dienste im Internet zur Verfügung.
Der Zweck der Dienste ist es, geographische Informationen oder geokodierte Daten als digitale Karten zu berechnen und diese, oder Informationen zu diesen, bereit zu stellen. Diese Daten stehen zumeist durch eine Datenbank zur Verfügung, können aber auch aus Dateien oder anderen Diensten gelesen werden. Die Karten werden zu Bildern berechnet bzw. gerendert, was entsprechend vorgenommenen Einstellungen auf dem MS erfolgt.
Die Erzeugung der Karten geschieht einzeln auf Anfrage oder im Vorfeld. Dabei ist es möglich, bei Anfragen Informationen mitzugeben, sodass die Karten entsprechend derer erzeugt und bereitgestellt werden.
In diesem Zusammenhang werden diese konfigurierten Karten Layer genannt. Layer stellen kategorisierte Teile des Datenbestandes dar und können gemeinsam eine Karte bilden. Es findet dabei eine Aufteilung in statische und dynamische Layer statt.
Statische Layer beinhalten immer die selben Daten.
Dynamische Layer enthalten dagegen nur von der Anfrage abhängige Daten, wobei eine unterschiedliche Darstellung möglich ist.
Dienste, welche der MS anbietet, sind selbst definiert oder OGC-konform.
Diese senden bei Anfrage Bilder oder Informationen der Layer an den Sender zurück. Notwendige Informationen der Anfragen sind die Koordinaten des Kartenausschnittes und der Maßstab. Weitere notwendige Informationen sind vom jeweiligen Dienst abhängig.
Zusätzlich besitzen die einzelnen MS Funktionen zur Umrechnung zwischen Koordinatensystemen oder zur individuellen Darstellung der Layer.
Die Speicherung erzeugter Bilder, wird von Tile Caches[5] realisiert. Diese sind eigenständige Programme oder direkt im MS integriert.
Dienste definieren die Kommunikation zwischen Client und Server und die Möglichkeiten des Renderns sowie Erhalts von relevanten Zusatzinformationen.

[4] [FÜ01] S. 36
[5] Definition unter Kapitel 3.3.1 S. 26

Die für diese Arbeit relevanten Dienste werden nachfolgend vorgestellt, wobei aber zunächst OGC erklärt werden muss.

Open Geospatial Consortium (OGC)
Das OGC ist eine gemeinnützige Organisation, welche im Sinne der Interoperabilität mehrere Standards, Studien und Modelle für raumbezogene Informationsverarbeitung veröffentlicht hat.
Diese Standards haben sich für die Verarbeitung von geographischen Daten durchgesetzt und werden von Geoinformationssystemen (GIS) implementiert.[6] Dazu zählen Web Services, welche besonders von MSn unterstützt werden. Diese Web Services dienen dem Austausch von Geodaten im Internet, sodass ein standardisierter, plattform- und sprachenunabhängiger Zugriff ermöglicht wird.[7]

Die nachfolgenden Dienste wurden vom OGC definiert.

Web Map Service (WMS)
WMS ermöglicht den Zugriff auf aus Vektor- und Rasterdaten generierten Karten und dazugehörige Informationen.
Der Client schickt einen HTTP Request an den Dienst, welcher unter einer bestimmten URL erreichbar ist. Dieser Request enthält GET Parameter, welche die Rückgabe des Dienstes definieren.
Dabei gibt es zwei grundlegende Arten der Rückgabe oder auch Operationen, welche durch den Wert des Parameters „REQUEST“ festgelegt werden:

- getCapabilities: liefert eine XML Datei, welche Informationen zum Service und allen Layern enthält. Diese Datei beschreibt zugleich alle nutzbaren Parameter. Bsp. einer Anfrage: `http://www.agriport.net/apps/nutrients/GFI?&REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.1.1`[8]
- GetMap: liefert eine Bilddatei der angeforderten Karte. Dabei werden Parameter, die aus der oben beschriebenen XML Datei enthalten sind, belegt mitgegeben. Bsp.: `http://www.agriport.net/apps/nutrients/GFI?LAYERS=Agricon:feldgrenzen&FORMAT=image/png&SRS=EPSG:900913&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&BBOX=1487158.82,6648186.97,1492050.79,6653078.94&WIDTH=256&HEIGHT=256`
  Wobei Parameter für den Layerausschnitt, den Namen des Layers oder die Größe des Bilders mitgegeben werden. Es besteht die Möglichkeit, mehrere Layer als eine Karte zu erhalten. Eine Erläuterung der Parameter, ist unter WMS getMap Paramter des jeweiligen MS's zu finden.

[6] vgl. [WIKI-OGC]
[7] vgl. [PR06] S. 36
[8] Der Prarameter „SERVICE“ bestimmt den zu nutzenden Dienst und „VERSION“ die Version des Dienstes

Abhängig vom MS und den vorgenommenen Einstellungen, ist es auch möglich mit dem Parameter „REQUEST=GetFeatureInfo“ Informationen zu einem Punkt zu erhalten oder mit „REQUEST=GetLegendGraphics“ die Legende des Layers als Bild zu erhalten. Die Legende wird aus zugewiesenen SLD Dateien erzeugt.
SLD Dateien werden mit Layern verknüpft, und beschreiben die Darstellung der Daten als Karte.

Durch die Übergabe von MS abhängigen speziellen Parametern ist es möglich, gekachelte Bilder des Kartenausschnittes abzurufen. Für jede Kachel wird eine Anfrage gesendet. Diese Kachelung kann ohne manuelle Eingriffe verwendet werden. Bei Nutzung dieser Kachelung spricht man von WMS Tiling Client Recommendation (WMS-C). Diese ist aber kein Standard des OGC, sondern noch in der Definition durch diverse Organisationen.[9]

Unter Kachlung versteht man die Teilung eines Kartenbildes in kleinere Bilder, Tiles genannt. Abbildung 2.1 soll der Beschreibung dienen:

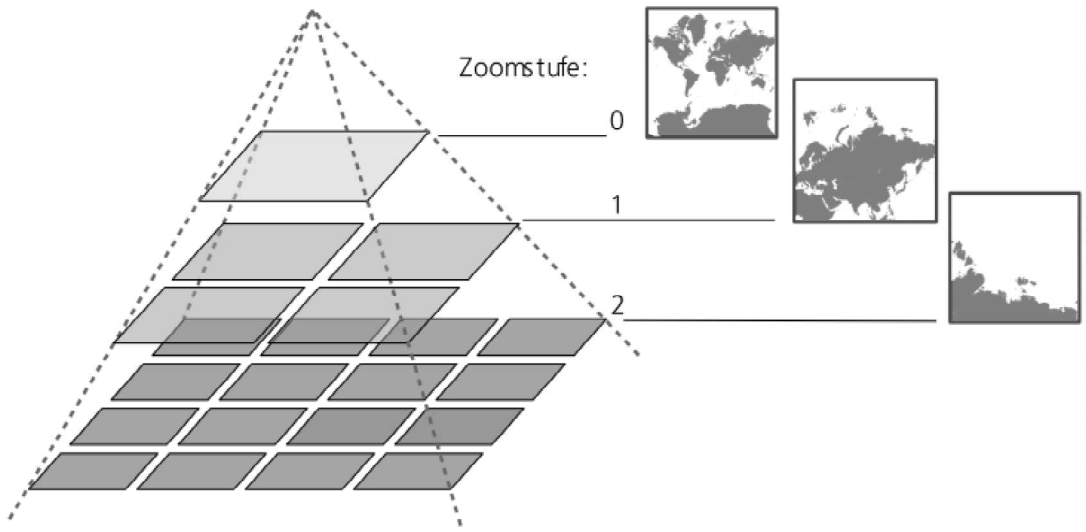

Abbildung 2.1: Kachelung einer Karte, von [JU10] Abbildung 2.3 Seite 12

Abhängig der Zoomstufe wird das Ausgangsbild und später einzelne Tiles in kleine Tiles aufgeteilt. Dies hat den Nutzen, dass bei einer Verschiebung der Kartenansicht nicht die gesamte Karte neu angefordert werden muss, sondern nur noch nicht erhaltene Tiles. Diese Aufteilung bzw. Ausdehnung der Tiles ist dabei vorgegeben.

Web Map Tile Service (WMTS)

WMTS besitzt viele Ähnlichkeiten mit WMS, wurde aber für eine schnellere und standardisierte Kachelung erstellt, ähnlich zu WMS-C.

[9] vgl. [OSGeo-WMSC]

Hierbei ist es dem Client möglich, mit HTTP und Get oder POST Parametern, sowie mit REST oder SOAP, einzig Tiles vom Server abzurufen.
Bei diesem Dienst gibt es 3 Operationen:

- GetCapabilities: liefert eine XML Datei, welche wie bei der gleichnamigen Methode von WMS den Dienst und die mitzugebenden Parameter beschreibt
- GetTile: liefert die Bilddatei einer einzelnen Kachel, wobei diese nur Daten eines Layers enthält. Es ist nicht wie bei WMS möglich mehrere Layer gleichzeitig mit einer Anfrage zu nutzen.
- GetFeatureInfo: analog zu WMS GetFeatureInfo

Nachfolgende Abbildung zeigt notwendige Prameter der HTTP GET GetTile Anfrage:

```
<message name="GetTileMessage_GET">
        <part name="service" type="wmts:RequestServiceType"/>
        <part name="request" type="wmts:GetTileValueType"/>
        <part name="version" type="wmts:VersionType"/>
        <part name="layer" type="xsd:string"/>
        <part name="style" type="xsd:string"/>
        <part name="format" type="ows:MimeType"/>
        <part name="TileMatrixSet" type="xsd:string"/>
        <part name="TileMatrix" type="xsd:string"/>
        <part name="TileRow" type="xsd:unsignedInt"/>
        <part name="TileCol" type="xsd:unsignedInt"/>
</message>
```

Abbildung 2.2: WMTS GetTile Abfrageparameter, Ausschnitt aus `http://schemas.opengis.net/wmts/1.0/wmtsAbstract.wsdl`

WMTS ist kombatibel zu TileCaches, welche mit dem MS verbunden werden und entsprechend einer Konfiguration Kacheln von Karten auf dem Server Zwischenspeichern, um bei Anfrage dieser Kacheln diese nicht erzeugen zu müssen.

Web Feature Service (WFS)

> „The OGC Web Map Service allows a client to overlay map images for display served from multiple Web Map Services on the Internet. In a similar fashion, the OGC Web Feature Service allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services.“[10]

WFS beschränkt sich dabei außschließlich auf Vektordaten und auf die Nutzung von HTTP als Netzwerkprotokoll. [11]

[10] [OGC-WFS] S. 12
[11] vgl. [GWIKI-WFS]

Jeder WFS Dienst muss folgende Operationen unterstützen:

> „GetCapabilities A web feature service must be able to describe its capabilities. Specifically, it must indicate which feature types it can service and what operations are supported on each feature type.
>
> DescribeFeatureType A web feature service must be able, upon request, to describe the structure of any feature type it can service.
>
> GetFeature A web feature service must be able to service a request to retrieve feature instances. In addition, the client should be able to specify which feature properties to fetch and should be able to constrain the query spatially and non-spatially.
>
> GetGmlObject A web feature service may be able to service a request to retrieve element instances by traversing XLinks that refer to their XML IDs. In addition, the client should be able to specify whether nested XLinks embedded in returned element data should also be retrieved.
>
> Transaction A web feature service may be able to service transaction requests. A transaction request is composed of operations that modify features; that is create, update, and delete operations on geographic features.
>
> LockFeature A web feature service may be able to process a lock request on one or more instances of a feature type for the duration of a transaction. This ensures that serializable transactions are supported.“[12]

Feature sind Abstraktionen von Objekten aus der realen Welt. Diese besitzen einen Namen, Attribute und geographische Informationen. Feature Types sind Gruppen von gleichartigen Objekten.
XLinks sind Verweise zwischen Einträgen in verschiedenen Dokumenten.

WFS ermöglicht einen umfangreichen Zugriff auf Daten und Darstellung dieser als Objekte.

[12] [OGC-WFS] S. 16

2.2 Ablauf AgriPort

AgriPort ist das umfangreiche Datenportal der Firma AgriCon GmbH. Diese Plattform ermöglicht eine automatische Verarbeitung und Darstellung der Precision Farming - Daten.

AgriPort (AP) diente dem Prototypen während des Entwurfs und der Implementierung als Vorbild. Dies soll bei der Erweiterung ebenso stattfinden, weshalb dessen Ablauf kurz vorgestellt wird.
Das Datenportal befindet sich derzeit in einer Umstellung des Mapservers. Bis dato wurde das kostenpflichtige Manifold System der Firma manifold.net verwendet. Dieses soll durch MapServer ersetzt werden.
Dafür wurde bereits eine prototypische Implementierung vorgenommen, welche nachfolgend dargestellt wird.

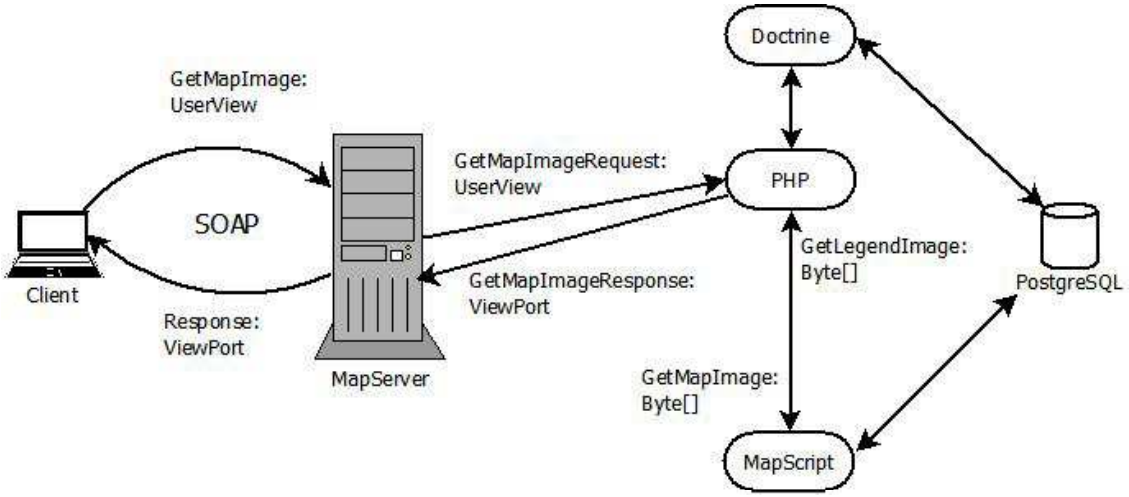

Abbildung 2.3: Ablauf AgriPort

Die Kommunikation zwischen Client und Mapserver findet über das Protokol SOAP statt. Dabei werden XML Dateien übertragen.
Aktuell wird bei jeder Verschiebung der Karte und Neuauswahl der Kartenelemente eine Anforderung an den Mapserver versandt.
Diese Anforderung enthält eine Struktur UserView, welche alle notwendigen Informationen wie die UserID, die Layer und die Ausdehnung der Karte enthält.
Diese Struktur wird Serverseitig mit der Skriptsprache PHP ausgewertet und ein ViewPort zurückgegeben. Der ViewPort enthält Informationen zu der angeforderten Karte sowie ein Bild der Karte und der Legende.
Die detailierte Darstellung der Strukturen ist im Anhang A (Seite 68) zu sehen.
Zu ergänzen ist, dass PHP die Daten mit dem Objektrelationaler Mapper Doctrine aus der Datenbank liest. Dieser wird in PHP eingebunden und direkt angesprochen. Weiterhin realisiert PHP den Dienst des Servers mit Hilfe des PHP Frameworks Zend.

Zend ist OpenSource und erweitert PHP um Web-Technologien und -Services, Suchfunktionen, PDF-Erstellung, weitere Möglichkeiten des Datenbankzugriffes und vieles mehr.

Es ist wichtig darauf hinzuweisen, dass kein Standard des OGC genutzt wird und kein Tilling stattfindet.
Diese Implementierung wird von AgriCon in Zukunft um weitere Funktionen ergänzt und in absehbarer Zeit für das Datenportal AP eingesetzt.
Im Vergleich sollen die Möglichkeiten der Einbindung in das AP System durch den jeweiligen Mapserver mit aufgeführt werden.

2.3 Ablauf Prototyp

Der Webclient „Agriport Mobile“ für mobile Endgeräte, soll durch diese Analyse verbessert und verschnellert werden. Dieser ähnelt dem Datenportal AgriPort und ist unter `http://www.agriport.net/apps/nutrients` zu erreichen. Der Client basiert auf JavaScript bzw. Sencha Touch und OpenLayers. Mit Sencha Touch wurde die Oberfläche und die Programmlogik erstellt und mit OpenLayers die Verwaltung und Anzeige der Karte mit den verschiedenen Layern realisiert.
Es stehen, je nach vorliegenden Daten nach der Anmeldung, verschiedene Layer zur Auswahl. Diese Layer sind auch einzeln auf dem Mapserver abgebildet, und werden mit WMS abgefragt. Dabei wird für den aktuellen Kartenausschnitt nicht ein Bild sondern mehrere Kacheln vom Server angefordert.
Als Koordinatenreferenzsystem wird der EPSG-Code 3857 bzw. 900913 genutzt. EPSG:3857 stellt eine genauere Referenzierung als EPSG:900913 dar, wobei aber beide miteinander kompatibel sind. Da die Daten in EPSG:4326 vorliegen, muss durch die Datenbank oder den Mapserver eine Umprojektion stattfinden.
Im Hinblick auf den genaueren Aufbau, der Funktionsweise und den Layern, sei auf den Praktikumsbericht des Autors zu dem Prototypen hingewiesen.[13]
WMS soll als Dienst genutzt werden, auch wenn AgriPort diesen nicht verwendet.
Die WMS Nutzung wird OGC-konform durch OpenLayers realisiert. Auch wird HTTP zur Kommunikation verwendet, nicht SOAP.
In diesem Zusammenhang wird angestrebt, den Programmcode nicht zu verändern und einzig die WMS Anfragen anzupassen. Die Methoden GetMap und GetLegendImage des WMS sollen genutzt werden.

[13] [JU12]

2.4 Testumgebung

Die genutzte Umgebung, in welcher die Konfigurationen vorgenommen und die Tests durchgeführt werden, wird hiermit vorgestellt.

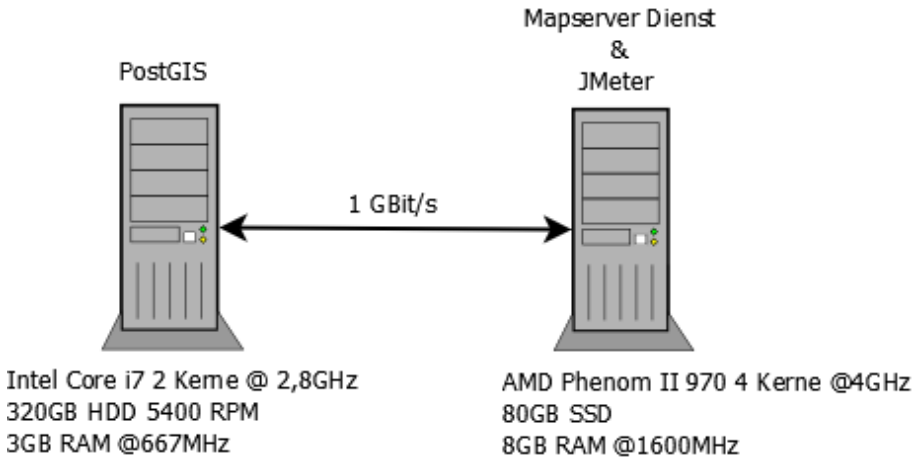

Abbildung 2.4: Testumgebung

Es findet eine Trennung zwischen Datenbank und Mapserver statt. Dies gewährleistet eine bessere Analyse und Vergleichbarkeit, da jeder Mapserver andere Datenbankabfragen nutzt. Es muss für eine objektive Beurteilung festgehalten werden, welcher Anteil der Request-Bearbeitungsdauer und des Aufwands auf den Mapserver und welchen auf die Datenbank zurückzuführen ist.

Auf beiden Computern läuft Windows 7 als Betriebssystem.
Dabei sind die Mapserver als Dienst installiert. Dies erlaubt die Ausführung als Systemprozess.

Der Testclient wurde mit dem Lasttest-Programm JMeter erzeugt.
Ein Lasttest ist ein Test einer Software, in welchem extreme Last auf das zu testende System einwirkt. Parallel zur Belastung findet eine Beobachtung des Verhaltens statt, sodass Fehler entdeckt oder Verbesserungsmöglichkeiten offen gelegt werden können.
JMeter ist ein freies in Java geschriebenes Programm, mit welchem über eine GUI Testabläufe definiert und ausgeführt werden können. Dabei können Schnittstellen wie HTTP, SOAP und JDBC angesprochen werden.

Die Definition und der Ablauf der Tests wird hier ebenfalls angegeben, da sie sich zwischen den Mapservern nicht unterscheiden.

Zweckmäßig dienen Tests zur Ermittlung der Ausführungsgeschwindigkeit oder Leistung. Für die Leistung eines Systems gibt es verschiedene Definitionen.

Zumeist ergibt sich die Leistung aus erfolgreichen Verarbeitungsvorgängen pro Zeiteinheit unter gegebenen Vorrausetzungen.

Da Software getestet wird, handelt es sich um Software-Benchmarks:

> „Software-Benchmarks dienen dazu, die Leistungsfähigkeit unterschiedlicher Programmiersysteme bezüglich der Ausführungsgeschwindigkeit zu vergleichen.
> Bei der Erstellung eines Software-Benchmarks wird derselbe Algorithmus in verschiedenen Programmiersprachen implementiert und die Ablaufzeiten der Programme untereinander verglichen.“[14]

Hier handelt es sich um Mapserver in Form von Software, deren Ablaufzeiten durch gleichbleibende Tests verglichen werden.
Die Aufgabe der hier vorgestellten Mapserver ist es, als Antwort auf WMS getMap Requests Bilder zurückzuliefern. Statt der Ablaufzeit muss folglich der Durchsatz bzw. die erfolgreich beantworteten Requests pro Sekunde gemessen werden.
Es handelt sich in diesem Fall speziell um Lasttests. Dabei wird zusätzlich die Systemauslastung und Fehlerquote festgehalten und als Vergleichskriterium genutzt.

Der Zweck der abschließenden Tests ist die, für den Prototypen spezifische, Leistung als Vergleichskriterium mit Hinblick auf die Systemauslastung aussagekräftig anzuführen.

In großen Geodatenverarbeitungssystemen werden Lasttests wie folgt durchgeführt:

> Abhängig der Anzahl der Clients wird die Systemauslastung und der beantworteten Requests pro Sekunde festgehalten. Es wird grob ein großer Bereich festgelegt, aus welchem zufällige kleine Bereiche als Karte abgefragt werden. Die Auswertung erfolg mit Hilfe von Tabellen und Graphen, welche die Messergebnisse wiederspiegeln.

Da in diesen Szenarien sehr viele Vektor- oder Rasterdaten vorhanden und in dem gewählten Bereich gleichmäßig verteilt sind, ist der zufällige Zugriff sinnvoll.
In den Tests dieser Arbeit werden die Schläge abgefragt. Diese sind nicht regelmäßig angeordnet, sondern bilden entfernt voneinander liegende Gruppen. Würde grob ein Ausschnitt gewählt, in welchem alle Schläge liegen, würden 90-97% aller Tiles keine Schläge beinhalten. Der Großteil aller Tiles wäre somit leer.
Da kein Client vorrangig leere Karten betrachtet und der Abruf dieser Tiles keinen Hinweis auf die tatsächliche Verarbeitunsdauer geben würde, werden in den Tests mehrere kleine Bereiche ausgewählt und abgefragt.

[14] [WIKI-Bench]

Der Testclient besitzt mehrere Threads, welche je einen Client darstellen. Es werden Test für 8, 16, 32, 64, 128 und 256 Clients durchgeführt. Jeder Client sendet 15 Requests pro Sekunde, was der durchschnittlichen Nutzung des Prototypen entspricht. Es liegen fünf Gruppen aus je 15 Tiles vor, welche zufällig von den Clients ausgewählt und abgefragt werden. Nach zwei Testdurchläufen werden 25 Sekunden gemessen und die Anzahl der beantworteten Request, die Fehlerquote, die Durchschnittsantwortzeit und die maximalen und minimalen Werte tabellarisch festgehalten.
Davor und während des Tests wird die Systemauslastung beider Computer festgehalten

Da Messungen einer gewissen Ungenauigkeit unterliegen, Messungen an sich das Messergebnis verändern, und zusätzlich diverse Hintergundprozesse Einfluss nehmen, geben die resultierenden Testergebnisse nur einen spezifischen und subjektiven Eindruck. Da AgriCon andere Computer mit anderen Betriebssystemen nutzt, wären die Ergbnisse dort andere, aber ähnliche.
Jedoch besitzen die Testergbnisse einen repräsentativen Charakter, wodurch sie für den Vergeich nutzbar sind.

3 GeoServer

3.1 Übersicht

GeoServer (GS) ist als Mapserver mit dem JAVA Server Jetty oder Tomcat verwendbar, kann aber auch als Dienst in einem Apache HTTP Server oder IIS eingebunden werden. Er beinhaltet sowie verwendet GeoTools. GeoTools ist ein OpenSource JAVA GIS Toolkit. Dieser Mapserver zeichnet sich durch besondere OGC Konformität hinsichtlich der verwendeten Standards und Dienste aus.
Die Konformität ist durch die Zusammenarbeit zwischen dem OGC und der Open Source Geospatial Foundation (OSGeo) bei Konferenzen, Dokumenten, besonderen Unternehmungen und öffentlichen Ereignissen begründet.[15] Durch GS werden folgende Dienste unterstützt: WMS, WMS-C, WMTS, WFS und WCS, sowie Standards wie Filter Encoding (FE), Styled Layer Descriptor (SLD) und Geography Markup Language (GML). GS ist freie Software der Open Source Geospatial Foundation und wird in dieser Arbeit in der als stabil gekennzeichneten Version 2.1.3 verwendet.

Die Konfiguration findet vor Allem durch die mitgelieferte GUI statt, welche es übersichtlich und mit geringem Aufwand ermöglicht, Datenquellen einzubinden und die Daten mit Diensten bereitzustellen. Ist in den nächsten Abschnitten von „der GUI“ die Rede, ist jene des GS gemeint. Nichtsdestotrotz wird in den nachfolgenden Kapiteln zusätzlich auf Konfigurationsdateien eingegangen. Diese befinden sich im Ordner „data_dir“, welcher bei der Installation des GS erstellt und angegeben wird.
Es besteht die Möglichkeit, erstellte Layer durch einen Klick mit OpenLayers darzustellen und somit direkt ohne Client auf Korrektheit zu überprüfen.
Bei der Datenabfrage aus externen Datenquellen kann der FE genutzt werden, was es beispielsweise erlaubt, eine parametrisierte SQL Abfrage mit Parametern aus der Clientanfrage durchzuführen. Dabei werden folgende Datenquellen unterstützt: [16]

- Vektordaten: Shapedateien, externe WFS, PostGIS, ArcSDE, DB2, Oracle Spatial, MySql und SQL Server
- Rasterdaten: GeoTiff, JPG, PNG, Bildpyramiden, GDAL Formate und Image Mosaic

Ein weiteres Feature ist die Umprojektion von Daten in Echtzeit, sodass diese mit der Projektion EPSG:4326 direkt vom Server in EPSG:900913 umprojeziert werden können. GS beinhaltet den TileCache GeoWebCache. Dieser ist ebenso in JAVA geschrieben und unabhängig des GSs verwendbar, ist aber für die Nutzung mit diesem optimiert.

[15] vgl. [OSGeo-OGC]
[16] vgl. [OSGeo-GS]

3.2 WMS

3.2.1 Einstellungsmöglichkeiten

Dieser Abschnitt zeigt die Konfigurationsmöglichkeiten mit WMS unter GeoServer auf.

Die Konfiguration kann über die mitgelieferte GUI oder manuell in den jeweiligen XML Dateien vorgenommen werden. Dabei unterscheidet sich die Anzahl und Art der Möglichkeiten kaum. Mit Hilfe der GUI ist eine Erläuterung der Möglichkeiten anschaulicher und wird deshalb im Nachfolgenden genutzt.

Um Daten als Karten per WMS verfügbar zu machen, müssen in folgenden Abschnitten Einstellungen und Eintragungen vorgenommen werden:

Abbildung 3.1: GeoServer Benutzeroberfläche

Arbeitsbereiche
In dieser Ansicht können Arbeitsbereiche erstellt, bearbeitet und gelöscht werden.
Ein Arbeitsbereich dient der Gruppierung von Layern und besitzt einen maximal 10 Zeichen langen Namen sowie eine optionale URI, über welche der Arbeitsbereich bei Verwendung von Namensräumen in einer Java Naming and Directory Interface (JNDI) Umgebung angesprochen werden kann. Dem Arbeitsbereich zugeordnete Layer werden über „Arbeitsbereich:Layername“ angesprochen.

Datenquellen
Hiermit ist es möglich, Datenquellen zu konfigurieren. Dabei unterscheiden sich die Datenquellen in Quellen für Vektor-, Rasterdaten und andere WMS-Dienste. Es ist möglich bestehende WMS oder WFS Dienste, Dateien und Datenbanken als Quellen zu nutzen. Siehe dazu 3.1.[17]

Der Prototyp verwendet PostGIS, was unter Vektordaten aufgelistet ist. Dafür werden die nachfolgenden Einstellungsmöglichkeiten angegeben.
Es muss der Name, die Zugangsdaten und der zuzuordnente Arbeitsbereich angegeben werden.
Zusätzlich können Einstellung für die Verbindung und die Verwendung der Datenbank vorgenommen werden, welche die Leistungsfähigkeit stark beeinflussen können.
Neben der Anzahl der Verbindungen zu PostGIS und dem Timeout in Sekunden einer jeden Verbindung, ist es auch möglich, vor jeder Benutzung einer Verbindung, diese mit „validate connections“ auf Konsistenz zu überprüfen.
Mit „Loose bbox“ wird direkt die bounding box, was das begrenzende Rechteck meint, der jeweiligen Geometrien genutzt, sodass die Geschwindigkeit enorm erhöht werden kann, aber geringe Ungenauigkeiten auftreten können. Dies ist nur mit WMS von Vorteil und sollte mit WFS nicht genutzt werden, zu große Ungenauigkeiten entstehen.[18]
„Estimated extends“ aktiviert die Nutzung des Befehls „SDO_TUNE.EXTENT“bei Datenbankzugriffen, welcher bei der Auslesung der bounding box eines Layers aus den angegebenen Daten genutzt wird. Bei Deaktivierung werden die gesamten Daten gelesen und ausgewertet, um die bounding box zu erhalten. Dies hat ergo nur Einfluss bei der Aktion des Auslesens der bbox.
Um gewisse Sicherheitsaspekte oder bestimmte Datentypen einzubeziehen ist nach den Tipps von Andrea Aime, technische Leiterin von GeoSolution, die Option „prepared-Statements“ vorteilhaft.[19] Nach Andrea Aime sollte dies nur verwendet werden, wenn es wirklich gebraucht wird. Weiterhin gibt es laut dem Unternehmen GeoSolutions eine negative Beeinflussung der Leistungsfähigkeit.[20]

[17] S. 15
[18] vgl. [GS-PG]
[19] vgl. [OSGeo-PG]
[20] vgl. [GeoS11] S. 34

Layer
In dieser Liste werden alle Layer zur Bearbeitung aufgelistet. Es ist es möglich weitere hinzuzufügen.
Dabei muss jeder Layer genau einem Arbeitsbereich zugeordnet werden.
Layer, welche ihre Daten aus PosGIS beziehen, erhalten ein SQL Statement zur Datenabfrage. Damit kann entsprechend den Zugangsdaten der Datenquelle jede beliebige Tabelle und jeder View abgefragt werden. Dieses Statement kann Funktionen der Datenbank und kartesische Produkte enthalten. Von der Verwendung von kartesischen Produkten ist abzuraten, da einige Funktionen vom GeoServer statt von der Datenbank ausgeführt werden könnten, was eine deutliche Verlangsamung der Datenverarbeitung mit sich bringt.
Während der Eingabe des Statements, ist es möglich dessen Rückgabeattribute mit Name und Typ anzuzeigen und für das geometry Attribut dessen Typ und die SRID zu setzen. Die SRID stellt eine Identifikationsnummer des Koordinatenreferenzsystems dar. Bei Verwendung des EPSG-Codes kann dieser direkt in SRID eingegeben werden. In jedem Fall empfielt die Dokumentation, vor der Speicherung des Statements, dessen primär-Attribut zu kennzeichnen.

GeoServer bietet die Möglichkeit, die Statements zu parametrisieren. Dabei wird aus der WMS oder WFS Anfrage der Parameter „viewparams“ ausgelesen, welcher mehrere weitere Parameter enthalten kann. Diese Parameter werden an das Statement weitergegeben. Der Syntax ist folgender: „parameter1:wert1;paremter2:wert2“
Die gewünschten Parameter werden über die GUI in der SQL View Sicht mit Name, Standardwert und prüfendem regulärem Ausdruck erstellt. Der Standardwert wird verwendet, wenn derjenige Parameter nicht im Parameter „viewparams“ enthalten war.
Der reguläre Ausdruck wird auf den ankommenden Parameter angewandt, sodass SQL Injections oder Fehleingaben entgegengewirkt werden kann. Die Verwendung im Statement erfolgt mit „%parametername%“.
Abbildung 3.2.1 veranschaulicht die Verwendung.

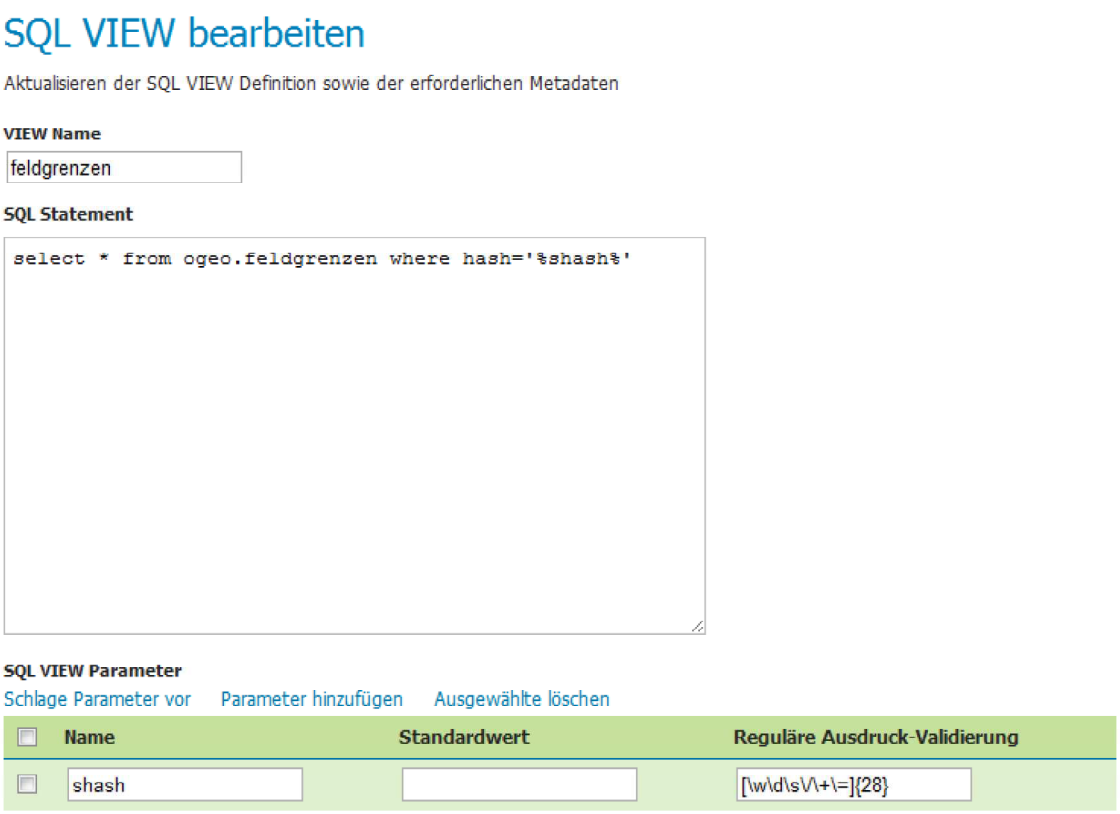

Abbildung 3.2: parametrisiertes SQL Statement

Nach der Angabe der Datenquelle ist der Name, Titel und die Beschreibung einzutragen. Zusätzlich können Angaben zur Katalogisierung angegeben werden.
Das Koordinatenreferenzsystem wird für die Quelle und für die Ausgabe eingestellt. Es ist möglich eine on-the-fly Umprojektion durch den GeoServer vornehmen zu lassen. Diese beansprucht aber wesentliche Rechenleistung, sodass eine Umprojektion vermieden werden sollte.
Weiterhin muss das begrenzende Rechteck (bounding box) eingetragen werden. Diese beschreibt die maximale Ausdehnung, in welcher sich die Daten befinden und sollte möglichst genau gewählt werden.
Bei keiner Verwendung eines parametrisierten Statements, kann das Rechteck direkt per Knopfdruck aus den Daten berechnet werden. Ansonsten muss dies manuell eingetragen werden.
Eine Inkorrekte Angabe der bbox schließt eventuell Daten aus.

In einem weiteren Reiter „Publizierung“ sind weitere wichtige Einstellungen möglich.
Bei GeoServer wird ein Layer über alle aktivierten Dienste automatisch bereitgestellt. Jeder Layer kann aktiviert und deaktiviert werden, wobei dazu eingestellt werden kann, ob dieser Layer in der GetCapability Anfrage der Dienste mit erscheinen soll oder nicht. Der Cache Header der Rückgaben kann in Bezug auf die zu cachende Zeit auf dem Client eingestellt werden.
Für WFS ist die maximale Anzahl der Features pro Anfrage und die maximale Anzahl

der Nachkommastellen der jeweiligen Feature-Werte einstellbar.
In Bezug auf WMS sind die verfügbaren Stile und der Standardstil aus den verfügbaren SLD Dateien des Ordners „data_dir/styles“ auswählbar. Außerdem kann der Standardwert des Buffers der Bildränder konfiguriert werden. Siehe dazu Parameter „buffer“ unter 3.2.2.[21]
Letztendlich sind Katalogisierungsangaben für WMS und Einstellungen für Keyhole Markup Language (KML) möglich. KML „ist eine Auszeichnungssprache zur Beschreibung von Geodaten für die Client-Komponenten der Programme Google Earth und Google Maps.“[22]

Gruppenlayer
Gruppenlayer fassen mehrere Layer zusammen. Dabei wird eine Rangfolge der Layer festgelegt, welche die Überlagerung der einzelnen Layer für die Kartenausgabe als Bild festlegt. Der Zweck ist die Möglichkeit mit einer Anfrage mehrere Layer zu nutzen,
ohne den Parameter „layers“ mit mehreren Argumenten zu belegen, sondern nur mit „Arbeitsbereich:Gruppenlayer“. Weiterhin werden alle Parameter an jeden einzelnen Layer weitergegeben.

Unter Layervorschau können statische Karten direkt und ohne Client mit OpenLayers angesehen werden.

Über die GUI ist es Zuätzlich möglich zu WMS, WFS und WCS Einstellungen vorzunehmen. Neben der Aktivierung und der Angabe verschiedenster optionaler Metadaten zum jeweiligen Dienst, können weitere Einstellungen vorgenommen werden.
Bei WMS ist es möglich folgende Einstellungen vorzunehmen:

- SRS Liste (EPSG-Codes) der getCapabilities Rückgabe definieren
- die Stufe des renderns von Rasterdaten festlegen (nächster Nachbar, Bilinear oder Bicubic, wobei „nächster Nachbar“ die schnellste und detailärmste Variante darstellt)
- Modien und Werte von KML
- Resourcenverbrauch (max. Speicher, max. Renderingzeit und max. Fehler) des Dienstes
- digitales Wasserzeichen, welches aktiviert und dessen URL, Transparenz und Position im Rückgabebild festgelegt werden kann
- Kompressionsraten von PNG und JPEG. In der genutzten Installation wurden PNG Bilder nicht komprimiert, sondern nur JPEG Bilder: Fehlermeldung: „Not compressing output for mimetype: image/png“, dies müsste durch neuere Versionen von GeoServer behoben werden.

[21] S. 22
[22] [WIKI-KML]

- Einstellungen des WMS-Animators, welche es ermöglicht statische Layer mit Animationen zu versehen.
- SVG Optionen für Art des Erzeugers und Antialiasing.

Dazu sollten alle nicht verwendeten Dienste abgeschaltet werden, da alle von Beginn an aktiviert sind. Sind nur die notwendigen Dienste aktiviert, ist der Mapserver entlastet und Clients haben keinen Zugriff auf Daten über nicht konfigurerte Dienste.

Um Sonderzeichen verschiedener Sprachen für Label und Bezeichnungen nutzen zu können, muss im jeweiligen SLD unter encoding in `<?xmlversion="1.0"encoding="ISO-8859-1"?>` der korrekte Zeichensatz angegeben werden.

Da es bei der Nutzung von Labels zu Schwierigkeiten in Hinsicht auf deren Positionierung und gegenseitige Beeinflussung kommt, müssen entsprechende Maßnahmen ergriffen werden.
GeoServer bietet dafür keine direkten Lösungen. Doch SLD stellt dafür allgemeine und GeoServer spezifische Optionen zur Verfügung.
Es ist somit möglich, die Position der Labels abhängig von ihrer Zuordnung zu bestimmen, sie zu rotieren, an Linien zu orientieren, zu gruppieren, Überschneidungen durch Prioriäten und Minimalabstände zu vermeiden und ein automatisches Anordnen zu aktivieren. Dabei sei auf die Dokumentation unter `http://docs.geoserver.org/2.1.3/user/styling/sld-reference/labeling.html` verwiesen.

Entsprechend diesen Einstellungsmöglichkeiten ist eine individuelle Konfiguration des WMS-Dienstes möglich.

3.2.2 WMS getMap Parameter

Tabelle 3.1 listet die mindestens mitzugebenden Parameter mit beispielhaften Werten auf:

Name	Wert
request	getmap
service	wms
version	1.1.1
format	image/png
width	340
height	480
srs	EPSG:900913
bbox	1473380,6587850,1476080,6591350

Tabelle 3.1: notwendige WMS getMapParameter

„version“und „service“werden bei nicht-vorhandensein entsprechend den anderen Parametern vom GeoServer eingesetzt, doch wird deren Mitgabe empfohlen.

GeoServer prüft auf zahlreiche weitere Parameter, welche in folgender Tabelle aufgelistet und beschrieben werden:[23]

[23] vgl. [GS-VP]

Parameter	Erklärung
angle	Rotiert die Karte um die angegebene Anzahl an Grad im Uhrzeigersinn. Nutzbar für jedes Rasterformat, PDF- und SVG-Ausgaben.
bgcolor	Farbe des Hintergrundes des Bildes.
buffer	Gibt die Anzahl der zusätzlich zu betrachtenden Pixel an, welche sich außerhalb der bbox des Requests befinden. Dies bezieht sich auf den Style, sodass jegliche Überlappungen von Elementen auch wirklich am Rand angezeigt werden. Beispielsweise könnte ein Punkt außerhalb der bbox einen 10 Pixel großen Kreis besitzen, welcher teilweise in die bbox hineinreichen würde. Mit einem buffer von 5 wird diese Überlappung dargestellt.
cqlfilter	Mit Common Query Language (CQL) ist es möglich die Daten mit der Anfrage zu filtern.
env	Darin enthaltene Parameter werden an das jeweilige SLD weitergegeben, sodass ein dynamisches styling möglich ist. [24]
featureid	Ermöglicht die Filterung mit „featurename.id“ der Daten entsprechend den Informationen aus der GetFeature Anfrage.
filter	Analog Parameter „cqlfilter“, benötigt aber CQL im XML-Format.
formatoptions	Erlaubt das setzen 3 wichtiger Eigenschaften der Rückgabe: antialiasing (nur für Rasterdaten; on, off, text), dpi (nur für Rasterdaten; standard 90) und layout (Zuweisung einer Dekorationslayout-Datei)
maxfeatures	Legt die maximale Anzahl der zu rendernden Features an.
palette	Ermöglicht die Verwendung einer bestimmten Farbtabelle, um die Bildgröße zu minimieren.
requestcharset	Gibt an, welchem Zeichensatz die Werte der Parameter entsprechen (in Version 2.1.3 nicht verwendbar)
styles	Zuweisung einer bestimmten SLD Datei, welche sich im Ordner data_dir/styles befindet.
sld	Selbe Funktion wie sldbody, nur das ebenso eine URL zur SLD Datei angegeben werden kann.
sldbody	Mitgabe von SLD-Code, anstatt Verwendung eines auf dem Server gespeicherten Styles.
startindex	Legt den Index fest, ab welchem die features gerendert werden soll. Setzt vorraus, dass die Datenquelle paging unterstützt.
tiled	Aktiviert oder deaktiviert die Kachelung.
tilesorigin	Bestimmt die untere linke Ecke des Kachelrasters. Notwendig für Metatiling (siehe 3.3.1) bei gleichzeitiger Verwendung mit OpenLayers im Client.
time	Bei Verknüpfung der Daten mit Zeitstempeln, wird mit diesem Parameter die dem Zeitstempel zugehörigen Daten gefiltert und geliefert.

Tabelle 3.2: GeoServer: zusätzliche WMS getMapParameter I

[24] vgl. [GS-SLD]

Parameter	Erklärung
transparent	Gibt an, ob leere Flächen zu 100% transparent sein sollen.
validateschema	Prüft den in sldbody mitgegebenen SLD-Code auf Korrektheit.
viewparams	Kann mehrere Parameter enthalten, welche direkt an den Layer weitergegeben werden. Notwendig für parametrisierte SQL-Views.

Tabelle 3.3: GeoServer: zusätzliche WMS getMapParameter II

Mit diesen Parametern können getMap Operationen gefiltert, mit Stilen versehen und ihr Aussehen verändert werden.

Ein verwendeter Request würde wie folgt aussehen: `http://localhost:8080/geoserver/wms?LAYERS=Agricon:feldgrenzen&FORMAT=image/png&SRS=EPSG:900913&TRANSPARENT=TRUE&TILED=true&VIEWPARAMS=shash:EeGP98dtbGVfRnMugaaadG29ZCw=&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&STYLES=&BBOX=1473380,6587850,1476080,6591350&WIDTH=340&HEIGHT=480`

3.2.3 WMS getLegendGraphic

Entsprechend den Regeln der zugehörigen SLD Datei wird mit dieser Anfrage eine Legende erzeugt und als Bild zurückgesandt. So enteht beispielsweise durch 12 Regeln folgende Legende:

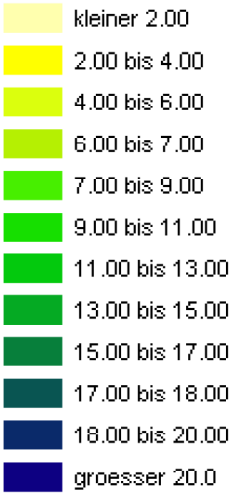

Abbildung 3.3: Geoserver getLegendGraphic Beispiel

Es werden die entsprechenden Rules[25] und FeatureTypes[26] untereinander mit deren Namen und einem farbigen Rechteck gezeichnet.

[25] Rules stellen Regels in SLD dar
[26] FeatureTypes stellen darstellbare Unterelemente von Rules dar

Diese Fähigkeit steht per sé zur Verfügung, und muss nicht aktiviert werden.

Für diese Anfragen können folgende Parameter genutzt werden:

Paramter	Erklärung
REQUEST	„GetLegendGraphic“
LAYER	Name des Layers
STYLE	zu verwendende SLD (optional)
FEATURETYPE	Angabe des zu nutzenden FeatreTypes des SLD (optional)
RULE	Angabe der zu nutzenden Rule des SLD (optional)
SCALE	bei Zoom-abhängigen Rules wird entsprechend der Zommstufe gerendert (optional)
SLD	siehe 3.3.2 (optional)
SLD_BODY	ibid
FORMAT	ibid
WIDTH	Breite der Rechtecke
HEIGHT	Höhe der Rechtecke
TRANSPARENT	true/false gibt an, ob der Hintergrund transparent ist (optional)
LEGEND_OPTIONS:	enthält Parameter, die das Aussehen der Legende bestimmen; key1:v1;key2:v2;...;keyn:vn (optional)
fontName	Name des zu verwendenden Schriftart
fontStyle	italic/bold
fontSize	Textgröße
fontColor	Textfarbe RRGGBB
fontAntiAliasing	true/false
bgColor	Hintergrundfarbe
forceLabels	on/off bestimmt ob Labels mitgezeichnet werden sollen

Tabelle 3.4: GeoServer: notwendige WMS getLegendGraphic

Ein Aufruf würde wie folgt lauten:
`localhost:8080/geoserver/wms?REQUEST=GetLegendGraphic&VERSION=1.0.0`
`&FORMAT=image/png&WIDTH=40&HEIGHT=20&LAYER=Agricon:biomasse`
`&STYLE=&TRANSPARENT=true&LEGEND_OPTIONS=fontSize:11;bgColor:F0F0F0;`

3.3 Erweiterungen

3.3.1 Tile Cache

Tile Caches (TC) dienen dem Speichern von Bildern, in diesem Zusammenhang auch Tiles genannt, auf dem Server, um das erneute rendern dieser zu vermeiden. Caches agieren zwischen Client und Mapserver. Die Anfragen der Tiles gehen an den Cache, dieser verarbeitet diese und fragt eventuell neue Bilder beim Mapserver ab.

Diese Tiles können durch Aufforderung im vornherrein, oder nur auf Anfrage erzeugt und gespeichert werden. Bei statischen Karten empfielt sich die Erzeugung und Speicherung im voraus, da dadurch kein Client auf die Erzeugung der Tiles warten muss, sondern diese direkt vom Speicher des Servers abgerufen werden. Man spricht in diesem Fall von seeding.
Für dynamische Karten, welche im Prototypen Anwendung finden, ist seeding nicht sinnvoll. Jeder Client fragt andere Kartenausschnitte mit dynamischen Informationen ab. Für jeden möglichen oder wichtigen Client müsste ein eigener seeding Prozess angestoßen werden und bei jeder Änderung der Daten per reseeding das Kartenmaterial aktualisiert werden. Folglich sollten Tiles auf Anfrage des Clients erzeugt werden. Dabei ist es möglich und notwendig die erzeugten Tiles mit einer Lebensdauer zu versehen, sodass das Kartenmaterial aktuell bleibt.
Die Lebensdauer sollte abhängig des Anwendungsfalles gewählt werden. Da für den Prototypen keine Erfahrungswerte vorhanden sind, sollen beim erreichen einer bestimmten Speicherplatznutzung die ältesten Tiles gelöscht werden.
GWC besitzt das Feature „disc quota“, welches die Löschung von alten Tiles bei überschreiten des festgesetzen Speicherkontigents automatisch ausführt. Dessen Konfiguration ist über die GUI des GeoServers möglich. Zusätzlich ist es möglich mit dem Attribut „expireCache“ die Dauer der Gültigkeit der Tiles festzulegen. Siehe dazu Listing 3.1.
Ist diese Dauer abgelaufen, wird der jeweilige Tile bei Anfrage neu erzeugt und gespeichert. Handelt es sich um Karten mit sich selten änderten Daten, besteht die Möglichkeit bei jeder Änderung der Daten diese Karten per Kommandozeile zu aktualisieren.
Der Tile-Cache des GeoServers bietet dafür eine REST-Schnittstelle, welche mit dem Tool cURL angesprochen werden kann.[27]
Damit ist es möglich seeding und truncating (Löschung) ausgwählter Karten durchzuführen und Karten zu verwalten.

GeoServer besitzt den integrierten Cache GeoWebCache (GWC) in der Version 1.3. Dieser ist auf die Nutzung mit GeoServer abgestimmt. Es wird WMS, WMS-C, WMTS, TMS, Google Maps KML und Virtual Earth unterstützt. Zusätzlich bietet er eine SOAP und REST-Schnittstelle zur Konfiguration und Handhabung der Tiles bei Verwendung von WMTS. Der GWC ist auch als Stand-Alone verfügbar und steht unter der GNU

[27] vgl. [GWC-REST]

Lesser General Public License (LGPL). Dazu ist eine GUI einzig für die Übersicht vorhanden, sodass Konfigurationen direkt in den jeweligen Dateien vorgenommen werden müssen.

Cachen von vorhanden Layern kann über zwei Wege aktiviert werden.
Es ist möglich die direkte WMS Integration über die GUI zu aktivieren, wobei alle aktivierten Layer automatisch durch den Cache gespeichert werden. Dabei ist es nicht möglich, Einstellungen für das Cachen der jeweligen Layer vorzunehmen. Weiterhin dürfen Parameter zur Filterung nicht verwendet werden. Über `http://localhost:8080/geoserver/wms` können die Layer nach der Aktivierung weiterverwendet werden. Wird die Íntegration nicht genutzt, kann für die jeweiligen Layer ein Eintrag in „data_dir/gwc/geowebcache.xml“ vorgenommen werden.
Ein Eintrag unter „<layers>“sieht wie folgt aus:

```
<wmsLayer>
   <name>Feldgrenzen</name>                          //Layername
   <mimeFormats>
       <string>image/jpeg</string>                   //Ausgabeformat
   </mimeFormats>
   <parameterFilters> //dynamische Parameter
       <regexParameterFilter>
           <key>VIEWPARAMS</key>
           <defaultValue></defaultValue>
           <regex>[\\w\\d\\s\\/\\+\\=\\:\\%]{34,100}</regex>
       </regexParameterFilter>
    </parameterFilters>
    <expireCache>600</expireCache>       //Gueltigkeitsdauer der Tiles in Sekunden
    <wmsUrl>
         <string>http://localhost:8080/geoserver/wms</string> //URL
    </wmsUrl>
    <wmsLayers>Agricon:feldgrenzen</wmsLayers>  //Layername im WMS-Dienst
    <transparent>true</transparent>              //Transparenz
</wmsLayer>
```

Listing 3.1: Layereintrag GWC

Die Koordinatenreferenzsysteme EPSG:4326 und EPSG:900913 sind dabei verwendbar. Um weitere verwenden zu können, müssen diese unter „<gridSets>“ definiert werden. Dafür ist die Definition eines Gridsets notwendig. Ein solches setzt sich aus dem begrenzenden Rechteck, dem Referenzsystem und Informationen zu den Tiles zusammen. Der Layer Feldgrenzen wird darauf über `http://localhost:8080/geoserver/gwc/service/wms` angesprochen.

In diesem Zusammenhang ist der Begriff Metatiling nennenswert. Hierbei erzeugt der GeoServer unabhängig von dessen Tile-Cache ein „Metatile“, welches aus mehreren Tiles besteht:

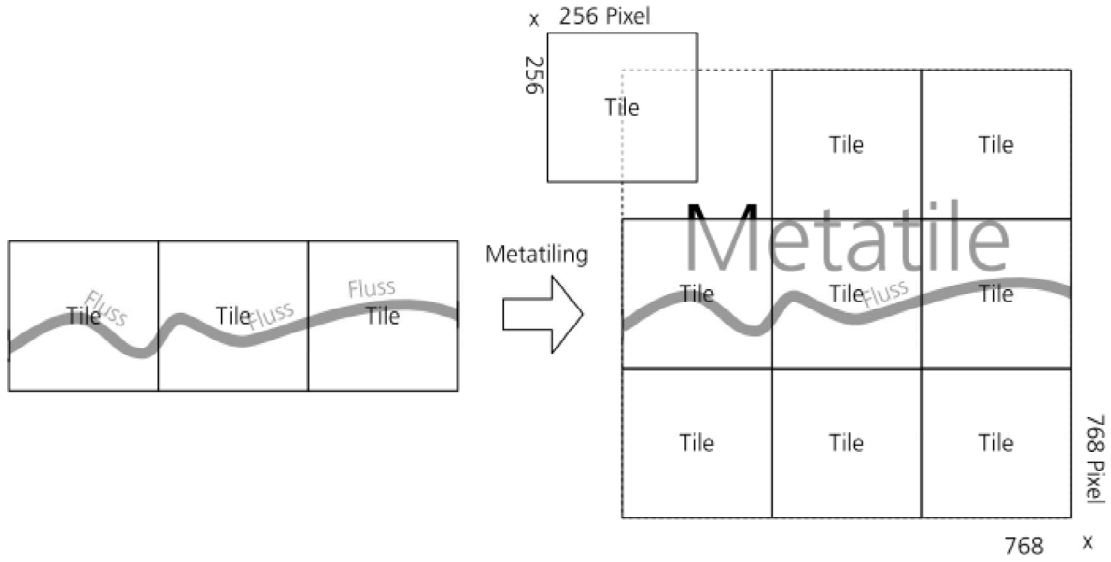

Abbildung 3.4: Metatiling, Bild von [JU10] S. 62

Dieses Metatile wird bei Anfrage eines enthaltenen Tiles erzeugt und kurzzeitig im Speicher gehalten. Werden nun Tiles eines bereits bestehenden Metatiles angefordert, wird das entsprechende Tile aus dem Metatile verwendet und kein neues erzeugt. Da dadurch die Gefahr besteht, dass viele Tiles der Metatiles nicht abgerufen werden, kann es durch die Erzeugung dieser gößeren Metatiles zu Leistungsminderungen kommen. Dies ist durch den größeren Speicherbedarf durch den Rendering-Prozess des größeren Metatiles begründet. Der Hauptvorteil des Metatilings liegt bei der Verwendung von Labels. Bei Tiles, welche direkt und nicht über ein Metatile abgerufen werden, kommt es bei Labeln zu Dopplungen und „Abschneidungen“, wie in Abbildung 3.4 zu sehen ist:

Abbildung 3.5: Doppelte Labels, Bild von `http://docs.geoserver.org/stable/en/user/_images/priority_lots.gif`

Da Tiles Teilbilder der Metatiles sind und die Erzeugung der Labels am Metatile vorgenommen wurde, reduziert sich damit der beschriebene Effekt.

Um Metatiling zu aktivieren, muss der Request folgende Schlüssel-Wert Paare enthalten:

- tiled=true
- width=265
- height=256
- tilesorigin=x,y (Koordinaten der untere linke Ecke der Karte des Clients)

Die Anzahl der in einem Metatile enthaltenen Tiles muss für alle Layer in der GUI oder in jedem Layer einzeln eingestellt werden. Individuel in layer.xml des jeweligen Layers. Die Einträge `<entrykey="GWC.metaTilingX">3</entry>` und `<entrykey="GWC.metaTilingY">3</entry>` bestimmten die Anzahl der horizontalen und vertikalen Tiles.

Die Verwendung von Tiles kann bei korrekter Nutzung die Leistungsfähigkeit des Mapservers erhöhen oder für eine bessere Darstellung der Karten genutzt werden. Der Prototyp verwendet dynamische Karten und keine Labels, weshalb zu diesem Zeitpunkt kein TC und kein Metatiling sinnvoll wären.
Um das erneute Abrufen der Bilder zu reduzieren, können diese mit einer Lebensdauer versehen werden, sodass sie Clientseitig im LocalStorage des Browsers gespeichert werden.

3.3.2 JAI

Java Advanced Imaging ist eine Java Bildmanipulationsbibliothek von Oracle. Mit dieser können komplexe Bildtransformationen durchgeführt werden, wobei die Bildformate BMP, GIF, JPEG, PNG, PNM und TIFF vielseitig unterstützt werden.[28]

Diese Bibliothek kann durch JAI Image I/O Tools erweitert werden, sodass das Lesen und Schreiben der Bilder beschleunigt werden kann.

[28] vgl. [WIKI-JAI]

In Verbindung mit GS wird die Bildverarbeitung wesentlich verschnellert:

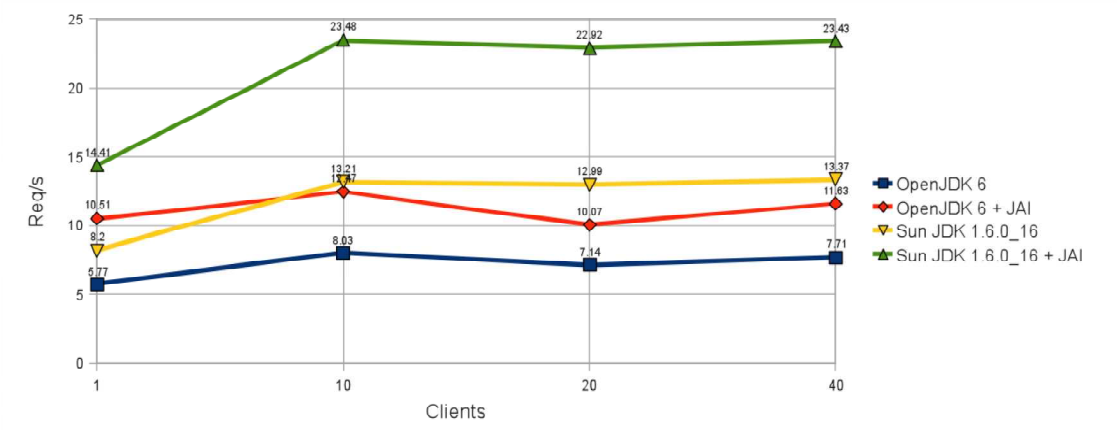

Abbildung 3.6: Beschleunigung durch JAI, Abbildung 10 S. 11 von [OG10]

Dieses Diagramm bezieht sich auf zufällige GetMap Anfragen mit dem Vektor-Layer TIGER roads, welcher standardmäßig in der GS Installation enthalten ist. In diesem Szenario steigt die Verarbeitungsleistung auf das 1,75-fache bei der Verwendung von JAI.

Eine solche Steigerung konnte nicht beobachtet werden. Einzig bei der Nutzung des GeoWebCaches konnte eine Beschleunigung und geringere Systemauslastung festgestellt werden. Die Beschleunigung durch JAI und JAI Image I/O ist stark vom Szenario abhängig, sollte aber immer installiert werden. JAI Image I/O sollte besonders bei Nutzung eines Tile-Cache verwendet werden, da ansonsten Einbußen in der Geschwindigkeit drohen.

JAI ist besonders für die Rasterdatenverarbeitung geeignet, bringt aber auch bei Vektordaten große Vorteile, da die daraus erzeugten Bilder selbst Raster sind.[29] Allgemein wird die Verwendung von OGeo und GeoSolutions empfohlen.

Im Fall GS wird durch JAI nativer Programmcode verfügbar, wodurch entsprechende Vorgänge weniger Ressourcen benötigen und schneller abgearbeitet werden.

Für Windows ist JAI nur in der 32-Bit Version verfügbar, sodass die gesamte JAVA Umgebung in 32 Bit installiert werden muss. 64 Bit bringt laut Oracle einen Leistungsnachlass, aber die Möglichkeit mehr als vier GB Arbeitsspeicher verwenden zu können.[30] JAI und JAI Image I/O muss über Oracle bzw. Java SUN heruntergeladen und installiert werden. Ein Neustart des GSs aktiviert die Verwendung dieser Bibliothek.

[29] vgl. [GS-JAVA]
[30] vgl. [O-FAQ]

Über die GUI des GS sind wichtige Einstellungen vorzunehmen:

- Der Anteil am der JVM zugeteilten Arbeitsspeichers für JAI
- Die Anzahl der maximal zu erzeugenden Threads
- Die Priorität der Threads
- Die Wiederverwendung von Tiles
- Die Verwendung von nativer JPEG und PNG Beschleunigung mittels spezifischem Code
- Die Nutzung nativer Mosaik Beschleunigung

Die native Beschleunigung von JPEG/PNG Bildern führte zu keiner messbaren Beschleunigung, einzig die resultierenden Bilder wurden größer. Folglich sollte diese Option nicht aktiviert werden.

3.3.3 serverseitige Anfrageauswertung

Durch die beschriebenen Einstellungsmöglichkeiten ist die serverseitige Anfrage-Auswertung nicht notwendig.

Um den Wunsch nach der Verwendung von aktuellen SLD's aus der Datenbank zu erfüllen, können eigene Skripte verwendet werden. Dazu soll die bereits auf dem Server verfügbare serverseitige Skriptsprache PHP verwendet werden. Auf Anfrage wird der dem Layer zugehörige Eintrag aus der Datenbank gelesen. Mit diesem Eintrag, welcher validiertes SLD darstellt, kann verschieden vorgegangen werden:

- Temporäre Speicherung der SLD-Datei auf dem Server. Der Client erhält eine zugehörige URL, welche er im „sld“ Parameter angibt.
- Rückgabe des gesamten Eintrages an den Client. Dabei müssen die Zeichen für HTTP aufbereitet werden. Der Client gibt dies anschließend im Parameter „sld-body“ mit.

Die zweite Möglichkeit ist am schellsten zu realisieren, da keine Rücksicht auf Schreibrechte auf dem Server genommen werden muss. Folgendes PHP Skript würde zur Anwendung kommen:

```
<?php
...
//Einbinden der Zugangsdaten und Tabellennamen
include 'data.php';

//beschaffen der POST-Variablen
$sessionHash = testAufSessionHash($_REQUEST["sessionHash"]);
$layername = testAufSemikolon($_REQUEST["layername"]);

//Verbindung herstellen
$conn_string = "host=".$host." port=".$port." dbname=".$dbname." user=".$dbuser." 
    password=".$dbpass;
```

```
$dbconn = pg_connect($conn_string);

//Query vorbereiten
$sql = 'select themexml from '.$dbtablelayerthemes.' where ';
$sql = $sql.'layername='.$layername;

//Resource anfordern
$res = pg_query($dbconn, $sql);

//Resource verarbeiten - besteht aus 1 spalte: themexml
$theme = pg_fetch_all_columns($res, 0);

//zuruecksenden (JSON)
echo '[{"layer": '.$layername.', "SLD": '.$theme.' }]';

//Verbindung trennen
pg_close($dbconn);
?>
```

Listing 3.2: Datenbankabfrage mit PHP

3.4 Leistung

3.4.1 Empfehlungen

Dienste

Nicht verwendete Dienste sollten abgeschaltet werden, um Ressourcen zu sparen und einen ungewollten Gebrauch von außerhalb zu vermeiden. Im Fall des Prototypen kann WCS und WFS abgeschalten werden. WFS bietet die Möglichkeit den Service Level auf basic einzustellen, wodurch nur lesende Zugriffe möglich sind und somit die Sicherheit erhöht wird, aber der Service verfügbar bleibt. Bei WMS sollten die Limits des Resourcenverbrauchs eingestellt werden. Genutzte Werte sind:

- Max. Rendering Speicher in Kilobyte (KB): 262144
- Max. Rendering-Zeit in Sekunden (s): 60
- Max. Anzahl Fehler beim Rendering: 1000

Als Bildformat sollte JPEG gewählt werden, da die Verarbeitungsdauer und Bildgröße im Gegensatz zu PNG in Tests positiver ausfiel. Die in 3.2.1[31] genannte Kompressionsrate von 50% sollte eingestellt werden. Da JPEG nicht verlustfrei ist, sollten Tests mit allen Layern durchgeführt werden, welche die Bildqualität überprüfen und als Ergebnis das beste Verhältnis aus Dateigröße und Bildqualität haben. Bei Nutzung des WFS Dienstes sollte die Anzahl und Genauigkeit der Features so gering wie möglich gehalten werden

PostGIS

In der PostGIS Datenquelle sollte „loose bbox“ aktiviert sein und nach GeoSolutions[32]

[31] S. 16
[32] vgl. [GeoS11] S. 34

„preparedStatements“ vermieden werden. Außerdem sollte bei Verwendung von Vektordaten in den jeweiligen Layern „geometry“ als Geometrytyp gewählt werden, da ansonsten Fehler auftreten können oder eine Umwandlung der Geometrien durch den GeoServer stattfindet.
Die Anzahl „max connections“ sollte der in der Datenbank eingestellten maximalen Anzahl an Verbindungen entsprechen. Der Schalter „validate connections“ prüft jede Verbindung vor deren Nutzung. Diese Sicherheit birgt eine Leistungseinbuße von bis zu 16%. Im Rahmen der Tests wurde dieser Schalter aktiviert.
Weitere Optimierungen können direkt in der Datenbank vorgenommen werden. Die Dokumentation von GeoServer weist darauf hin, dass die Indizierung der jeweiligen Tabellen erfolgt sein muss. Zuätzliche Optimierungen der Datenbank sind möglich und von Vorteil, aber nicht Inhalt dieser Arbeit.

GeoServer und Java

Java sollte als JDK (schließt JRE mit ein) installiert werden, damit extra Parameter wie „-server -Xms2048m -Xmx2048m -UseParallelGC -XX:MaxPermSize=128m“ in wrapper.conf angehangen funktionieren (wird von der Dokumentation des GeoServers, OpenGeo und GeoSolutions empfohlen). Diese Parameter ermöglichen die Nutzung von mehr Arbeitsspeicher, des parallelen Garbage Collectors und eines größeren Heap Speichers.
Im Ordner „data_dir“ kann eine Datei „controlflow.properties“ abgelegt werden, wodurch das Control Flow Modul aktiviert wird. Mit diesem können die maximalen Anfragen pro Client und die maximale Anzahl an gleichzeitigen getMap Anfragen bestimmt werden.

Bei Nutzung und Anpassung an den jeweiligen Server ist eine Leistungssteigerung möglich:

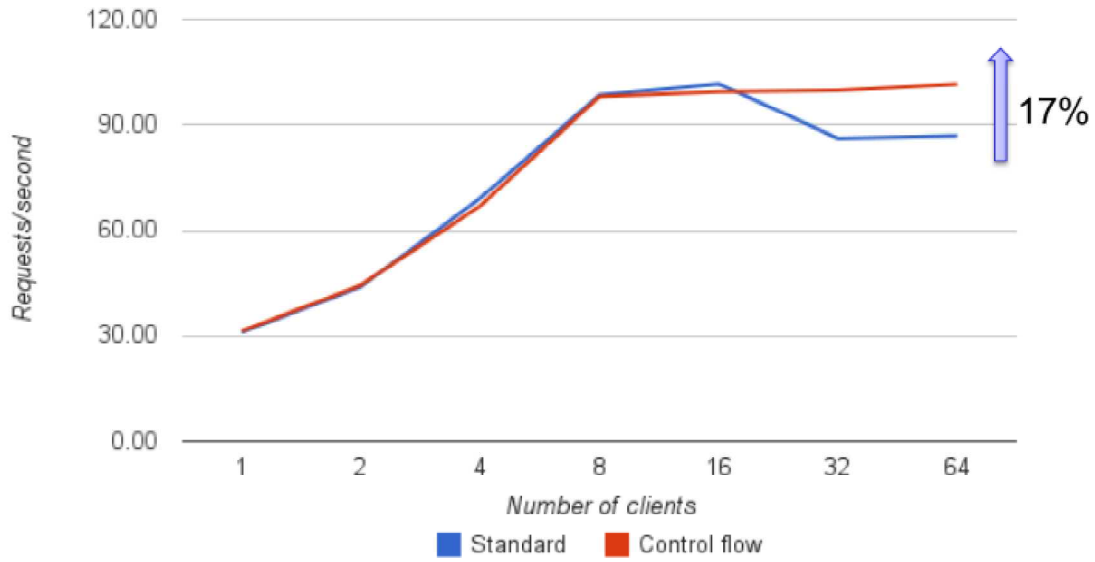

Abbildung 3.7: Control Flow Modul, Ausschnitt aus [GeoS11] S. 60

In der verwendeten Version war dieses Modul nicht verwendbar, sollte aber bei nachfolgenden Versionen funktionieren und benutzt werden.
Die maximale Anzahl dieser Anfragen sollte mit der in der Datenquelle eingestellten Anzahl der Verbindungen gleich sein. Wie in Abschnitt 3.3.2 gezeigt wurde, ist die Installation von JAI mit den genannten Vorkehrungen von Vorteil.
Einen besonderen Einfuss hat das Logging auf die Performanz. Es sollte nur so wenig wie notwendig geloggt werden, da ansonsten starke Leistungseinbußen drohen. GeoServer besitzt fünf Arten des Loggens, wobei „PRODUCTION“ einzig Fehler loggt und für den Betrieb empfohlen wird. Diese Arten können in den Dateien „Name.properties“ im Ordner „logs“ individualisiert werden.

Von GeoSolutions und OpenGeo wird die Empfehlung gegeben, mehrere Instanzen des GeoServers parallel zu nutzen.[33] Dadurch wird die Verfügbarkeit erhöht, sofern die Anfragen gleichmäßig an die jeweiligen Instanzen verteilt werden. Dazu ist es möglich einen WatchDog zu verwenden, welcher regelmäßig den Status einer jeder Instanz prüft. Sofern eine Instanz keine Rückmeldung mehr liefert, wird diese beendet und neu gestartet. Doch dieser Aufbau ist für hoch verfügbare Server mit mehr als 30000 Anfragen pro Sekunde geeignet und eignet sich für den Prototyp nicht. Bei dem Prototypen fallen derzeit schätzungsweise höchstens 750 Anfragen pro Sekunde an:

- jeder Client verschiebt oder zoomt im Mittel 1 mal pro Sekunde die Karte, wobei jeweils etwa 15 Anfragen gesendet werden

[33] vgl. [GeoS11] S. 66 und [OG10] S. 7

- wahrscheinlich nicht mehr als 50 Clients gleichzeitig aktiv

Eine solche Aufteilung auf Instanzen wäre möglich, indem GeoServer mit einem Apache mit dem Tool „mod_balance“ verwendet wird. „mod_balance“ erlaubt nicht nur Anfragen zu verzögern, sondern auf verschiedene Instanzen eines Dienstes zu verteilen.[34]

Caching

Tile Caches sollten wenn möglich genutzt werden. Für den Prototypen bietet es sich an, die Rückgaben der GetMap Anfragen mit einer Lebensdauer zu versehen, sodass sie auf dem Client gespeichert werden.
Sollte der Prototyp um Layer erweitert werden, welche statisch sind, oder selten aktualisiert und von vielen Clients verwendet werden, ist der Einsatz des Tile-Caches GeoWebCache sinnvoll. Der TileCache GeoWebCache ist in GeoServer automatisch aktiviert und sollte aus Gründen der Performance extra abgeschalten werden, sofern er nicht genutzt werden soll.
Dieser ist in der GUI im Menüpunkt GeoWebCache abzuschalten. Außerdem in den jeweiligen Layern unter „data_dir/workspaces/Arbeitsbereich/Datenquelle/Layername“ in layer.xml unter metadata folgender Eintrag vorzunehmen:
`<entrykey="GWC.enabled">false</entry>`
Damit wird sichergestellt, dass der Layer nicht vom Tile Cache verarbeitet wird.
In diesem Zusammenhang kann Metatiling genutzt werden, wodurch Tiles kurzzeitig auf dem Server gespeichert werden und die Abfragen der Datenbank vermindert werden. Metatiling kann zusammen mit normalem Tiling stattfinden, um die beschriebenen Effekte mit Labeln zu vermeiden. Dabei muss aber beachtet werden, dass nicht doppeltes Metatiling stattfindet, da GWC und GeoServer Metatiling unterstützen. Bei Verwendung von Metatiles empfiehlt sich die Größe der Metatiles an die Abrufe der Clients anzupassen. Für das in 2.4[35] beschriebene Verhalten der Clients bietet es sich an Metatiles mit fünf horizontalen und drei vertikalen Tiles zu verwenden.

Styling

Jedes SLD sollte einen FeatureTypeStyle enthalten, und nicht mehr. Jeder FeatureTypeStyle allokiert seine eigene Zeichenoberfläche.[36] Bei vielen Elementen sollte die zu rendernde Anzahl von der Zoomstufe abhängig gemacht werden.

[34] vgl. [1h-mb]
[35] S. 12
[36] vgl. [GeoS11] S. 38

3.4.2 Tests

Zum direkten Vergleich der Leistungsfähigkeit, wurden drei Tests entsprechend den unter 2.4[37] beschriebenen Bedingungen durchgeführt und dokumentiert. Gleichzeitig wurden die im vorherigen Punkt beschriebenen Empfehlungen angewandt.

Der erste Test nutzte weder den Tile Cache GeoWebCache, noch Metatiling.

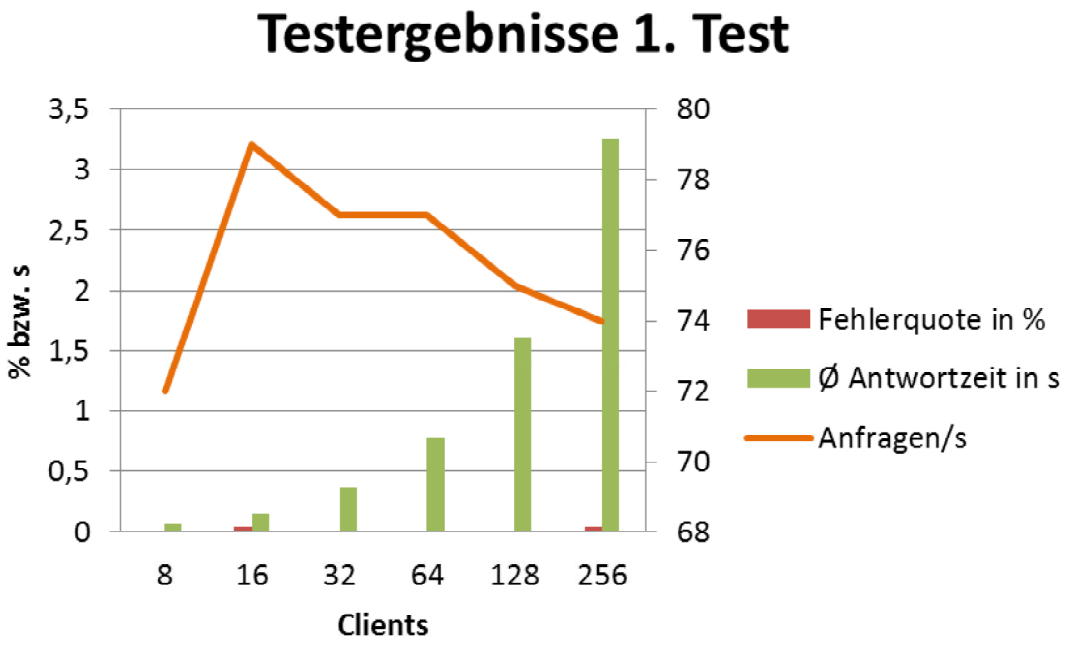

Abbildung 3.8: GeoServer 1. Test

Dieses Ergebniss ist zufriedenstellend. Bei Antwortzeiten unter einer halben Sekunde, ist die Verzögerung kaum bemerkbar. Bei mehr als 100 Clients steigt die Antwortdauer jedoch auf über einer Sekunde. Sollte der Prototyp eine derartige Nutzung erfahren, muss über Maßnahmen zur Lastverteilung nachgedacht werden.

[37] S. 12

Die Systemauslastung war dabei auf beiden Systemen gering:

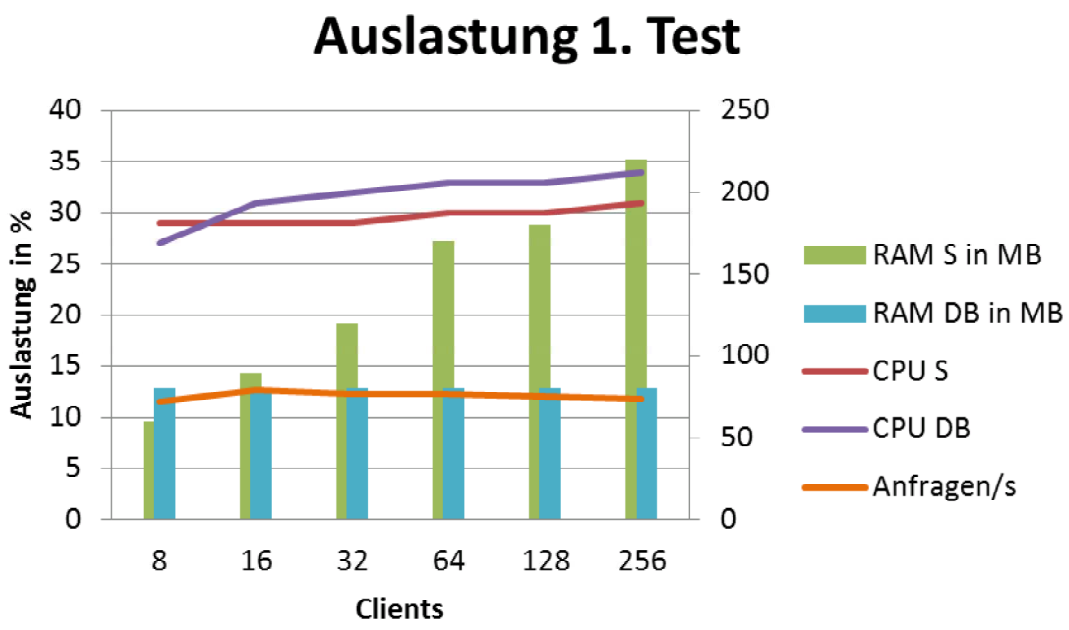

Abbildung 3.9: GeoServer Auslastung 1. Test

Der zweite Test nutzte Metatiling.
Das Ergebnis ist dem des ersten Tests sehr ähnlich:

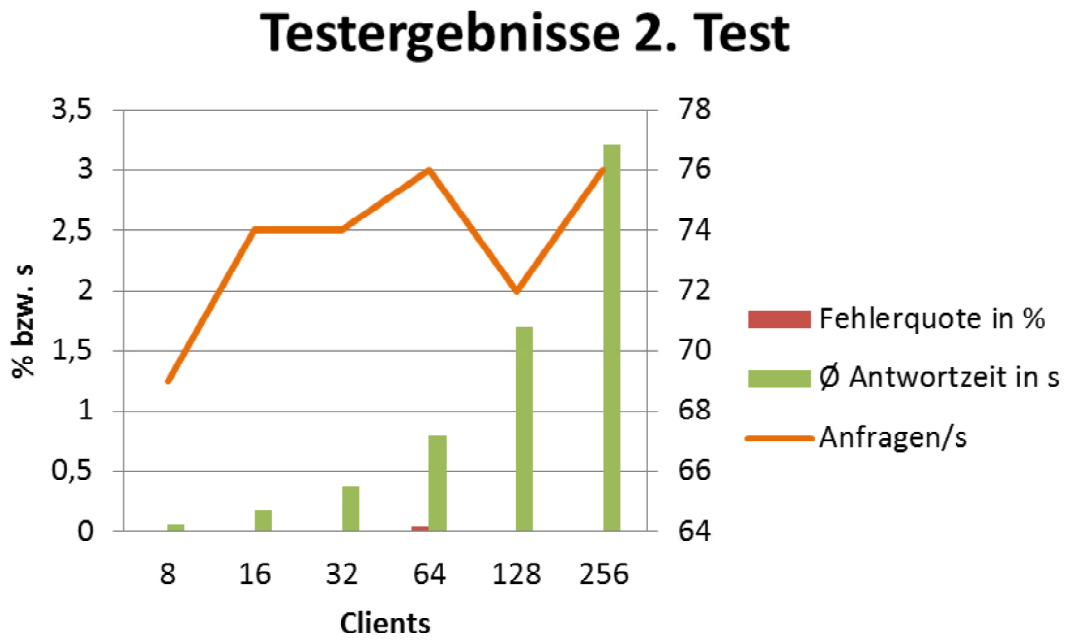

Abbildung 3.10: GeoServer 2. Test

Ebenso die Systemauslastung:

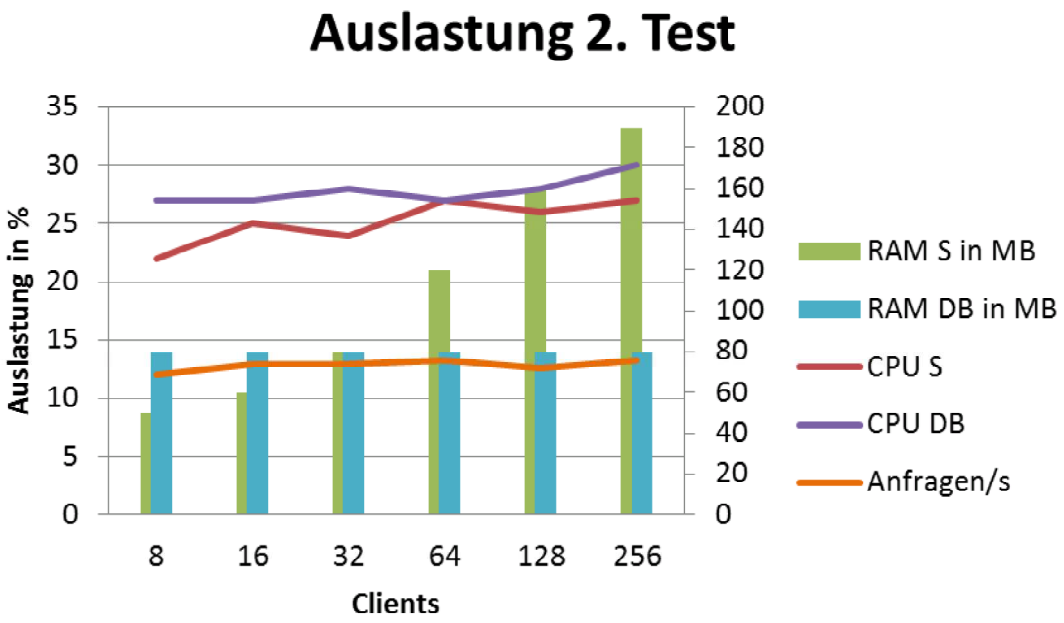

Abbildung 3.11: GeoServer Auslastung 2. Test

Diese Ergebnisse zeigen auf, dass Metatiling die Leistungsfähigkeit bei Verwendung des Prototypen nicht beeinflusst. Somit wäre die Nutzung von Labels und Metatiling unproblematisch in Hinsicht der Leistungsfähigkeit des Mapservers.

Der dritte Test nutzte den GWC.

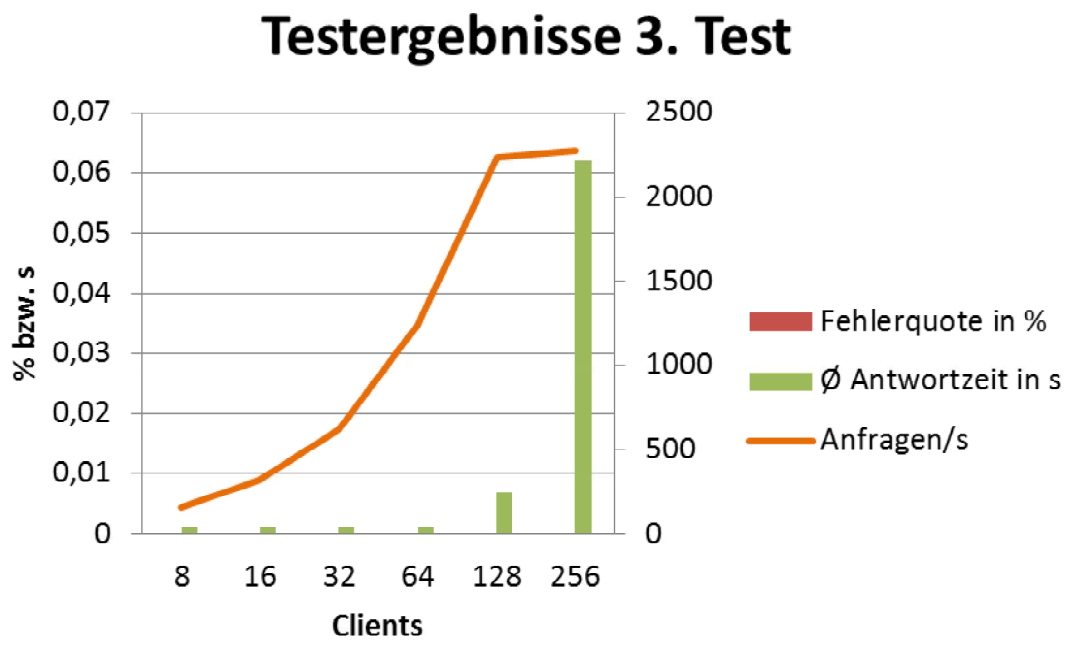

Abbildung 3.12: GeoServer 3. Test

Trotz der größeren Anzahl an beantworteten Requests pro Sekunde, ist die Systemauslastung geringer als in den anderen Tests:

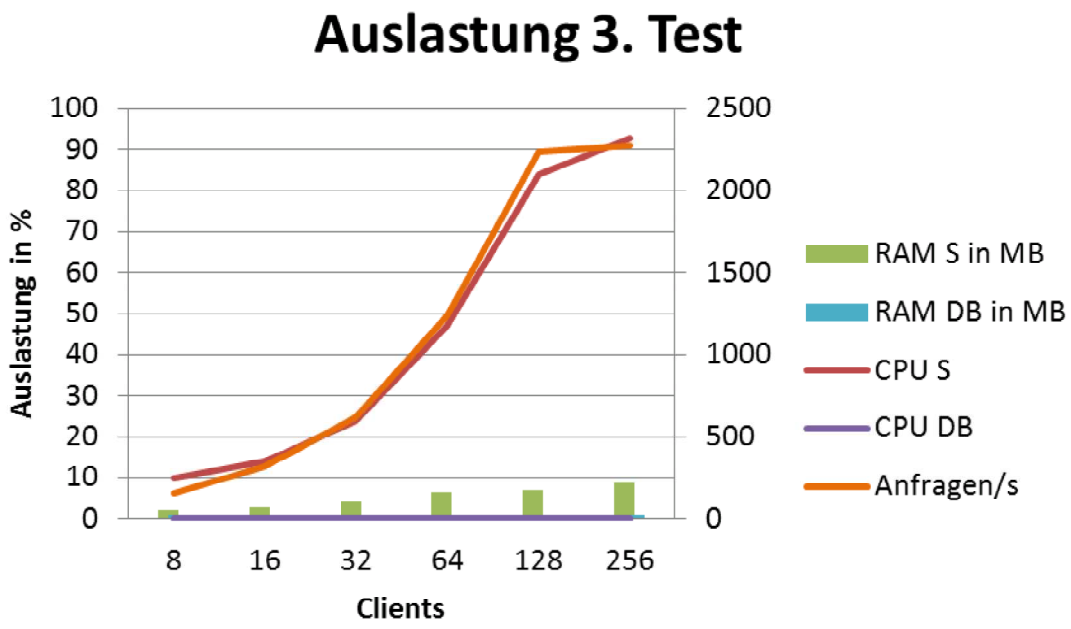

Abbildung 3.13: GeoServer Auslastung 3. Test

Dieses Ergebnis zeigt, wie sinnvoll die Nutzung eines Tile-Caches sein kann. Sind die jeweiligen Tiles einmal erzeugt, werden diese in wenigen Millisekunden zurückgesendet. Lediglich die erste Anfrage eines Tiles wird in der im ersten Test zu sehenden Zeit beantwortet.

Wird im GeoServer ein ausführliches Logging aktiviert, sinkt die Leistung drastig. Aus diesem Grund kann keine Angabe des Datanbankanteiles an der Verarbeitungsdauer gegeben werden. Die Tests wären ansonsten extrem verfälscht. Jedoch kann angegeben werden, dass GeoServer pro Anfrage mehrere Datanbankabfragen durchführt. Vor der Datenabfrage werden Informationen über die Datenbank abgefragt.

Die niedrige Auslastung des Geoserver Systems deutet auf ein Problem mit der JVM oder Latenzprobleme der Datenbankverbindung hin. Im Laufe der Thesis konnte dieses Problem nicht gelöst werden.

4 MapServer

4.1 Übersicht

MapServer, oder auch UMN MapServer genannt, ist ein in der Programmiersprache C geschriebener Mapserver, welcher frei zur Verfügung steht. In dieser Thesis meint Mapserver den unter 2.3[38] beschriebenen allgemeinen Mapserver und MapServer diesen speziellen Mapserver.

> „MapServer was originally developed by the University of Minnesota (UMN) ForNet project in cooperation with NASA, and the Minnesota Department of Natural Resources (MNDNR). Later it was hosted by the TerraSIP project, a NASA sponsored project between the UMN and a consortium of land management interests.“[39]

Und ist nun ein freies Projekt der OSGeo.

Die Dienste des MapServer können verschieden genutzt werden:

CGI-Skript
MapServer wird als Common Gateway Interface (CGI)-Skript auf einem bestehenden HTTP-Server verwendet. Dabei wird dieses Skript über eine URL verfügbar, wodurch Anfragen verarbeitet werden können.
Durch GET oder POST Parameter werden Informationen für die Anfrageverarbeitung mitgegeben. Ein Pflichtparameter „MAP“ gibt ein Mapfile an. Dieses Mapfile beinhaltet die Definition einer Karte, mit Layern, Informationen zur Karte und weitere Konfigurationselemente. Dieses wird abgearbeitet, sodass bei Angabe einer Template Datei interaktive Karten auf HTML-Seiten als Rückgabe erfolgen. Außerdem sind damit ohne Template OGC-konforme Dienste nutzbar: WMS, WFS und WCS

MapScript

> „Die MapScript-Bibliothek enthält Funktionen des Mapservers, z.B. das Lesen von Map-Dateien, das Rendern von Karten usw. Mit dem fertigen Skript können Karten generiert und in einem Browser angezeigt werden.“[40]

Somit ist es möglich mit der gewählten Programmiersprache eine ankommende Anfrage auszuwerten, um mit MapScript die notwendigen Informationen auszulesen und Bilder zu

[38] S. 11
[39] [MS-A]
[40] [PR06] S. 53

erstellen, welche anschließend zurückgesendet werden. Damit ist eine sehr tiefgreifende individuelle Erstellung eines eigenen Dienstes möglich.
Die Nutzung eines Mapfiles ist optional.
MapScript ist direkt und indirekt (über SWIG) in den Programmiersprachen C, C++, PHP, Python, Ruby, Java, Perl und C#/.NET verfügbar.[41]

Durch CGI und MapScript ist MapServer plattformunabhängig.
Die zu nutzenden Elemente müssen jedoch auf dem Server kompiliert werden. Da ein starker Zusammenhang zwischen den einzelnen Modulen und Bibliotheken besteht und zusätzlich entsprechende Kompiler für C und Python in dem gewählten Betriebssystem bereit stehen müssen, wurde für diese Arbeit auf eine bestehende Umgebung für Windows zugegriffen.
Diese nennt sich „MS4W“ , ist unter `http://www.maptools.org/MS4W/` verfügbar und beinhaltet unter anderem einen Apache HTTP Server, PHP 5.4.3, MapServer als CGI Modul und MapScript für CSharp, Java, PHP und Python. Damit sind die hier vorgestellten Anwendungsmöglichkeiten und Erweiterungen unter Windows möglich und nutzbar.
Einzig die Apache Konfigurationsdatei httpd.conf muss in Bezug auf die Umgebung verändert werden.[42]

Als Datenquellen können standardisierte Formate und Geodatenbanken verwendet werden:[43]

- TIFF/GeoTIFF, EPPL7 und GDAL
- ESRI Shapdateien, PostGIS, ESRI ArcSDE, Oracle Spatial, MySQL und andere

MapServer unterstützt ebenso on-the-fly Umprojektionen. Es steht keine GUI zur Verfügung, ergo müssen alle Einstellungen manuell in den jeweiligen Dateien vorgenommen werden.

[41] vgl. [WIKI-MS]
[42] Siehe dazu 4.2.1 S. 43
[43] vgl. [MS-A]

4.2 WMS

4.2.1 Einrichtung

Die Einrichtung erfolgt sehr unterschiedlich, je nach System. Hierbei wird das unter 4.1[44] beschriebene MS4W als Grundlage verwendet.

Es gibt 2 Möglichkeiten, um Karten durch den WMS Dienst auszuliefern:

MapScript
Mit Hilfe von MapScript ist es möglich, einen OGC-konformen WMS zu erstellen.
Zwar kann ein WMS-konformes Mapfile eingelesen werden, doch die Weitergabe der Anfrageparameter und deren Überprüfung muss mit MapScript bzw. in der verwendeten Programmiersprache erfolgen.
Kapitel 4.3[45] geht auf diese Thematik ein.

CGI-Skript mit Mapfile
Die Nutzung des CGI-Skriptes ist weniger aufwendig. Da das CGI-Skript WMS-konforme Parameter direkt für die Verarbeitung des Mapfiles nutzt, ist für WMS neben dem CGI-Skript einzig ein WMS-konformes Mapfile bereitzustellen. Ein solches Mapfile liegt im Anhang A[46] bei.
Damit ein Mapfile WMS-konform ist, müssen nach der Dokumentation folgende Elemente definiert sein:

> „At the MAP level:
> Map NAME
> Map PROJECTION
> Map Metadata (in the WEB Object):
> wms_title
> wms_onlineresource
> wms_srs (unless PROJECTION object is defined using „init=epsg:...“)
> wms_enable_request
>
> And for each LAYER:
> Layer NAME
> Layer PROJECTION
> Layer METADATA
> wms_title
> wms_srs (optional since the layers inherit the map's SRS value)
> Layer STATUS

[44] S. 41
[45] S. 49
[46] S. 74

Layers set to STATUS DEFAULT will always be sent to the client."[47]

Dabei sind einige Elemente optional bzw. redundant, da Layer Eigenschaften des MAP-Objektes erben.

Eine Einführung in den Aufbau eines Mapfiles, ist im Anhang C[48] zu lesen.

Da bei der „MS4W "-Umgebung das MapServer CGI-Skript nicht aktiviert ist, müssen in „httpd.conf"der Apache Konfiguration folgende Einträge eingefügt werden:

```
<Directory "/ms4w/Apache/htdocs">
        Options All
        AllowOverride None
        Order allow,deny
        Allow from all
</Directory>
<Location "/cgi-bin">
        Options ExecCGI
        SetHandler cgi-script
        Order allow,deny
        Allow from all
</Location>
<IfModule mime_module>
        TypesConfig conf/mime.types
        AddType application/x-compress .Z
        AddType application/x-gzip .gz .tgz
        AddHandler cgi-script .cgi .exe
        AddType application/x-httpd-php .php5 .php4 .php3 .phtml .php
        Action  application/x-httpd-php /cgi-bin/php-cgi.exe
</IfModule>
```

Listing 4.1: httpd.conf Eintrag für CGI

Dabei müssen sich die Anfragen an „localhost:84/cgi-bin/mapserv.exe" richten.

MapServer ist ebenso als Fast-CGI-Skript verfügbar.
Fast-CGI stellt einer Weiterentwicklung von CGI dar und ist besonders durch eine höhere Verarbeitungsleistung gekennzeichnet.

[47] [MS-WMS]
[48] S. 74

Bei Nutzung von Fast-CGI konnte eine Halbierung der Verarbeitungsdauer beobachtet werden:

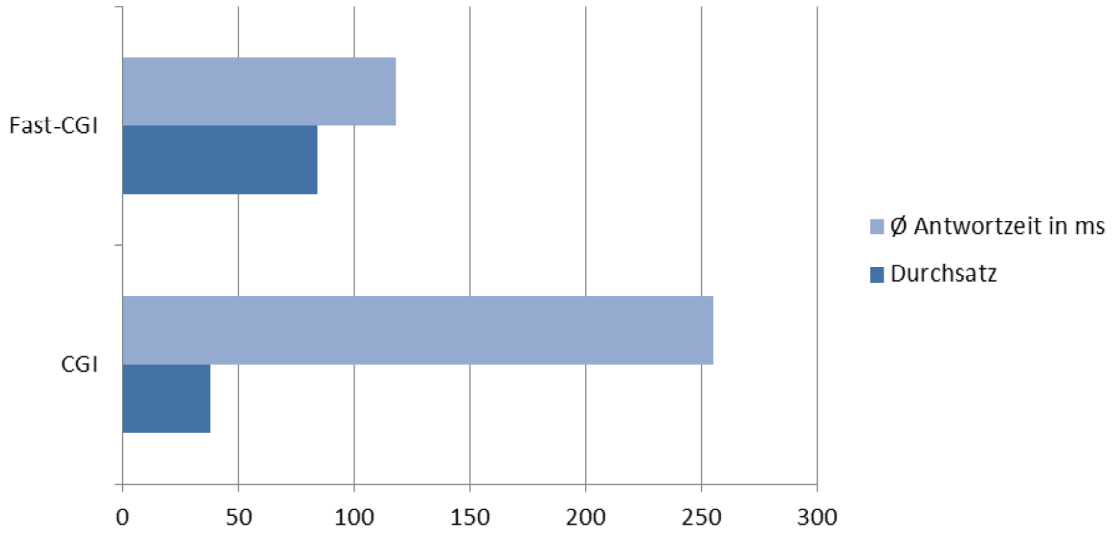

Abbildung 4.1: CGI und Fast-CGI Performance

Um Fast-CGI nutzen zu können, müssen erneut Einträge in „httpd.conf“ vorgenommen werden:

```
LoadModule fcgid_module modules/mod_fcgid.so
<IfModule fcgid_module>
        FcgidMinProcessesPerClass 0
        FcgidIdleScanInterval 1
        FcgidProcessLifeTime 10
</IfModule>
```

Listing 4.2: httpd.conf Eintrag für Fast-CGI

Der Pfad ändert sich dabei in:
„localhost:84/fcgi-bin/mapserv.exe“

Auf Grund der WMS Konformität und Leistungsfähigkeit, wird das Fast-CGI Skript in den abschließenden Tests verwendet.

4.2.2 WMS Parameter

MapServer behandelt Anfragen sehr strikt, wodurch eine sofortige Fehlerausgabe bei nicht übergebenen Parametern erfolgt. Nachfolgend werden die Parameter benannt.

GetMap
Folgende Parameter sind pro Anfrage mindestens notwendig:

Parameter	Erklärung
Map	relativer Pfad zur Mapdatei
Version	Versionsnummer des Dienstes
Request	Art der Anfrage
Layers	Layernamen
Styles	bevorzugter Style (muss in Mapdatei vorhanden sein - Parameter SLD/SLD_BODY kann auf SLD verweisen/SLD übergeben)
SRS	EPSG Bezugssystem
BBOX	Ausdehnung begrenzenden Rechtecks des Bildes
Width	Breite des Bildes
Height	Höhe des Bildes
Format	Bildformat

Tabelle 4.1: MapServer: notwendige WMS getMap-Parameter

GetLegendGraphic
Um GetLegendGraphic zu nutzen, muss das jeweilige Mapfile folgende Objekte enthalten:

- LEGEND im MAP Objekt
- CLASS in jedem LAYER-Objekt
- NAME und STATUS=on in jedem CLASS-Objekt

Die in der Anfrage mitzusendenden Parameter sind:
MAP, REQUEST, LAYER, FORMAT, WIDTH und HEIGHT, sowie optional SLD, SLD_BODY, SLD_VERSION, SCALE, STYLE und RULE.
Beispiel:
`http://localhost:84/fcgi-bin/mapserv.exe?map=WMSmap.map&SERVICE=WMS&VERSION=1.1.1&layer=Ph-Verteilung&REQUEST=getlegendgraphic&FORMAT=image/png`

4.2.3 Styling

Neben SLD ist mit dem Objekt CLASS in einem Mapfile ein individuelles Styling möglich. Besonders bei der Verwendung von Labeln, bieten sich viele Konfigurationsmöglichkeiten.
Im Objekt CLASS können direkt die Farbgebung, Größe, Umrandung und Transparenz angegeben werden. Zusätzlich kann mit dem Objekt LABEL im Objekt CLASS Labeling angewandt werden.
Darin können verschiedenste Parameter festglegt werden: [49]

- ANGLE
- BUFFER
- FORCE („Forces labels for a particular class on, regardless of collisions. Available only for cached labels. Default is false. If FORCE is true and PARTIALS is false, FORCE takes precedence, and partial labels are drawn.“)
- MAXLENGTH
- MAXOVERLAPANGLE ()
- MAXSIZE „(Maximum font size to use when scaling text“)
- MINDISTANCE
- MINFEATURESIZE („Minimum size a feature must be to be labeled.“)
- MINSIZE („Minimum font size to use when scaling text“)
- PARTIALS („Label could run off the edge of the map“)
- POSITION
- PRIORITY
- REPEATDISTANCE

Mit SLD ist es möglich, die Position der Labels abhängig von ihrer Zuordnung zu bestimmen, sie zu rotieren, an Linien zu orientieren und mit <![CDATA[<Code>]]> Zeilenumbrüche und andere Zeichen entsprechend dem Zeichensatz einzufügen.

4.2.4 Einbindung in AgriPort

Der prototypische Entwurf des MapServers für das Datenportal AgriPort wird auf kurz oder lang für den Produktiveinsatz genutzt werden.
Der Entwurf unterliegt der ständigen Verbesserung und Anpassung, trotz dessen soll im nachfolgenden aufgezeigt werden, welche Möglichkeiten es gibt, den Entwurf für die Nutzung von Agriport Mobile zu erweitern.

[49] vgl. [MS-L]

Vier Möglichkeiten bieten sich an:

1. Nutzung eines selbst erstellten PHP-Skriptes, welches MapScript nutzt. Ein solches wird in 4.3[50] erstellt. Der Vorteil liegt in dessen Kürze. Eine Nutzung des vorhandenen Entwurfes findet dabei aber nicht statt.
2. Da die direkte Nutzung des CGI-Skriptes per URL Sicherheitskritisch ist, kann mit Hilfe eines Wrapper-Skriptes ein Aufruf verarbeitet und weitergegeben werden.
   Die besondere Verbesserung der Sicherheit ist dabei, dass Serverseitig keine Ausführung von „.exe“ und „.cgi“ erlaubt werden muss.
   Nachfolgend ein Beispiel, welches unter Linux anwendbar ist:

```
#!/bin/sh
MAPSERV="/path/to/my/mapserv"
MAPFILE="/path/to/my/mapfile.map"
if [ "${REQUEST_METHOD}" = "GET" ]; then
  if [ -z "${QUERY_STRING}" ]; then
        QUERY_STRING="map=${MAPFILE}"
  else
        QUERY_STRING="map=${MAPFILE}&${QUERY_STRING}"
  fi
  exec ${MAPSERV}
else
  echo "Sorry, I only understand GET requests."
fi
exit 1
# End of Scrip
```

   Listing 4.3: Wrapper Skript, von [MS-W]

   Unter Windows muss das Skript entsprechend angepasst werden.
3. Anfertigung eines PHP Skriptes, welches die HTTP Anfrage aufnimmt, aus dieser ein UserView-Objekt[51] erzeugt und dieses an den SOAP-Server, bzw. eine Unterklasse der Serverinstanz weiterreicht.
   Anschließend muss das Bild extrahiert und an den Client gesendet werden.

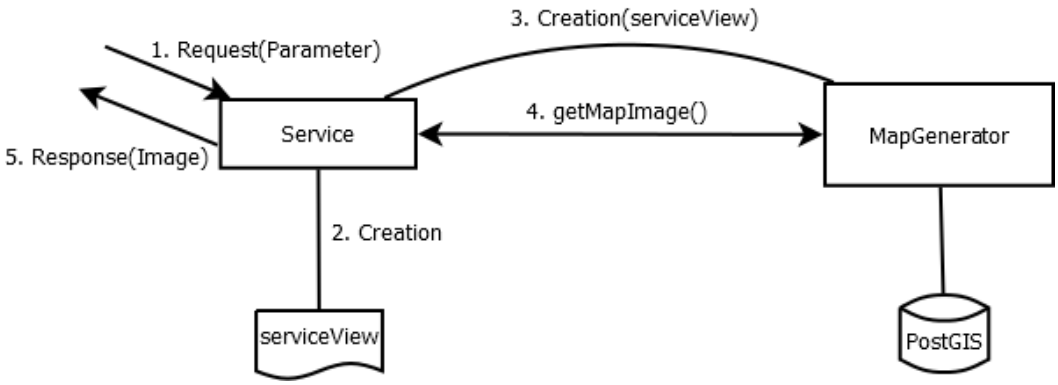

Abbildung 4.2: MapScript PHP Service

[50] S. 49
[51] Siehe SOAP Klassen S. 68

4. Zusätzlich zur SOAP-Schnittstelle eine JSON-Schnittstelle einbinden. So ist es möglich, den vollen Umfang des Entwurfes zu nutzen. Diese Schnittstelle kann für weitere Anwendungen, welche nicht SOAP nutzen können, verwendet werden. Doch der Entwurf und die Einbindung der Schnittstelle erfordert einen hohen Arbeitsaufwand serverseitig und Anpassungen clientseitig.

4.3 MapScript

Dieser Abschnitt gibt Informationen zu Mapscript und zeigt an Hand eines Beispiels dessen Nutzungsmöglichkeit.

Wie bereits in 4.1[52] erwähnt, bietet MapScript Funktionen und Funktionalitäten des Mapservers aus einer Programmiersprache wie PHP, Python, Ruby, Java, Perl oder C#/.NET herraus.
Es ist damit die Erstellung eines eigenen Dienstes möglich.
Bereits vorhandene Mapfiles können direkt ausgelesen und weiterverwendet werden. Somit ist es sinnvoll, ein Muster mit den wichtigsten Einstellungen abzulegen und wiederholt zu verwenden.
Bezüglich des Anwendungsfalles Agriport Mobile ist die Erstellung eines WMS-konformen Dienstes notwendig.
Dazu wird PHP genutzt, da es bereits eingerichtet und verfügbar ist:

```
<?php
//Parameter auslesen
//Parameter: BBOX, SRS, Width, Height, Format, Map, Service, Request, Layers,
    Styles, Transparent und shash
//fehlt ein Parameter, ist einer nicht WMS konform oder entspricht nicht einer WMS
    getMap Anfrage, wird eine Fehlermeldung zurueckgegeben.
$service = $_REQUEST["SERVICE"];
if (0 != strcasecmp($service, 'WMS')){
        fehler('falscher Dienst');
        return;
}
$version = $_REQUEST["VERSION"];
if (0 != strcasecmp($version, '1.1.1')){
        fehler('falsche Version des Dienstes');
        return;
}
$request = $_REQUEST["REQUEST"];
if (0 != strcasecmp($request, 'GetMap')){
        fehler('einzig GetMap Request moeglich');
        return;
}
$map = $_REQUEST["map"];
if (0 != strcasecmp($map, 'WMSmap.map')){
        fehler('nicht zugelassenes Mapfile');
        return;
}
$layer = $_REQUEST["LAYERS"];
if (0 != strcasecmp($layer, 'Feldgrenzen')){
        fehler('nicht vorhandener Layer');
        return;
```

[52] S. 41

```
}
$format = $_REQUEST["FORMAT"];
if (0 != strcasecmp($format, 'image/png')){
        fehler('falsches Bildformat');
        return;
}
$srs = $_REQUEST["SRS"];
if (0 != strcasecmp($srs, 'EPSG:3857')){
        fehler('falsches Referenzbezugsystem');
        return;
}
$transparent = $_REQUEST["TRANSPARENT"];
if (0 != strcasecmp($transparent, 'true')){
        fehler('Transparenz ist notwendig');
        return;
}
$bbox = $_REQUEST["BBOX"];
$width = $_REQUEST["WIDTH"];
$height = $_REQUEST["HEIGHT"];
$style = $_REQUEST["STYLES"];
$shash = $_REQUEST["shash"];

//vorhandenes Mapfile einlesen
$map = new mapObj("C:\ms4w\Apache\htdocs\PHPmap.map");

//Parameter setzen
$map->setSize($width, $height);
//bbox vorbereiten
$bbox_array = explode(',', $bbox);
$map->setExtent($bbox_array[0],$bbox_array[1],$bbox_array[2],$bbox_array[3]);
$feldgrenzen = $map->getLayer(0);
$feldgrenzen->setFilter('hash=\''.$shash.'\'');

//Bild erzeugen
$bild = $map->draw();

//Header erzeugen
header('Content-type: image/png');

//Bild ausgeben
$bild->saveImage();
?>
```

Listing 4.4: MapScript WMS Service

Dieser Ausschnitt zeigt exemplarisch, wie ein solcher Dienst funktionieren kann.
WMS-konforme Parameter werden ausgelesen, auf Korrektheit überprüft, bei einer korrekten Anfrage das Mapfile ausgelesen, entsprechende Parameterwerte eingesetzt und das gewünschte Bild erzeugt.
Jedes Objekt in einem Mapfile ist als Klasse mit Attributen und Funktionen in MapScript abgebildet. Dadurch ist nicht nur eine dynamische Erstellungen eines Mapfiles auf Anfrage möglich, sondern eine umfangreiche Programmierung eines eigenen Dienstes.

4.4 Tile Caches

MapServer besitzt keinen eingebauten Tile Cache (TC).
Es stehen aber mehrere Systeme zur Verfügung. Die bekanntesten sollen hiermit vorgestellt werden.
Da der Prototyp und AgriPort nicht auf einen TC angewiesen sind, wird nur oberflächig auf diesen Punkt eingegangen und einzig die Möglichkeiten genannt.

Alle Systeme haben folgende Punkte gemeinsam:

- Unterstützung von Metatiling
- die Konfiguration findet über ein oder mehrere Textdateien statt
- in diesen Konfigurationsdateien werden Einstellungen zu den Grundinformatioen, den Layern, den Bildformaten und den Gridsets vorgenommen
- Verarbeitung von GetFeatureInfo Anfragen möglich
- Speicherung der Tiles auf der Festplatte, dem Arbeitsspeicher oder in einer Datenbank
- Anstoßen von angepasstem Seeding möglich

TileCache
TileCache steht unter der BSD Lizenz und wird von der Firma MetaCarta Labs entwickelt. Dieser Cache eignet sich für die Dienste WMS-C und TMS. Außerdem wird Python zur Ausführung benötigt.
Der Zugriff erfolgt mit Hilfe eines (Fast-)CGI-Skriptes und speziell für den Apache mit mod_python. TileCache befindet sich derzeit in Version 2.11.
Für MS4W gibt es eine veraltete Beta Version, welche aber nicht mit der verwendeten Version des MapServers nutzbar ist.
Auf der Homepage `http://tilecache.org/` kann der Python Quellcode heruntergeladen werden, welcher bei der ersten Ausführung kompiliert wird. Dieser wird in den „cgi-bin“ Ordner des Apache kopiert und über „tilecache.cgi“ angesprochen. Die Konfiguration erfolgt in der Datei „tilecache.cfg“, welche sich im selben Ordner wie das CGI-Skript befinden muss.
TileCache ermöglicht nur die Weitergabe von OGC-konformen Parametern, sodass weder „map“ oder „shash“ zum MapServer durchgereicht werden können. Der Layertyp „MapServerLayer“ ermöglicht die Nutzung des „map“ Paramters, „shash“ kann aber nicht weitergegeben werden.
Um dieses Problem zu lösen, sind Veränderungen am Quellcode notwendig.

MapProxy
MapProxy wird von Omniscale entwickelt und steht unter der Apache Lizenz 2.0. Die Dienste WMS-C, TMS und WMTS mit KML Ausgabe werden unterstützt.
Die Homepage `http://mapproxy.org/` wirbt mit folgenden besonderen Fähigkeiten:

Reprojektion von Quelldaten, in Form von Daten und Bildern
interne und externe Bilder können hinsichtlich des Bildformates geändert werden

MapProxy benötigt ebenso Python zur korrekten Ausführung, da er als Python Programm über die Kommandozeile gestartet wird.
Die allgemeine Konfiguration erfolgt in der Datei „mappproxy.yaml“ und das Steuern von Seedingprozessen in der Datei „seed.yaml“. Beide Dateien sind im YAML Format, was „eine vereinfachte Auszeichnungssprache (engl. markup language) zur Datenserialisierung“[53] ist. Auch hier ist es nicht möglich, weitere Parameter wie „shash“ an MapServer weitergeben zu lassen. Diese Möglichkeit ist aber für die kommende Version 1.5 angekündigt.[54]

MapCache

MapCache befindet sich derzeit im BETA Status und wird in Zukunft mit MapServer als Paket angeboten werden. Dieser TC wird mit Leistungsfähigkeit und schneller, sowie einfacher, Installation geworben. WMS-C, TMS, WMTS und Virtual Earth sind dabei nutzbar.
Zu den besonderen Fähigkeiten zählt eine dynamische Parameterweitergabe, Komprimierung und Vervielfältigung von Tiles und die Fähigkeit, Tiles als zusammengesetzte Bilder zurückzusenden. MapCache ist als (Fast-)CGI-Skript einsetzbar.
Für MS4W steht kein Download zur Verfügung.
Somit muss dieser TC selbst kompiliert werden. Da dies aber verschiedene Bibliotheken mit bestimmten Versionen vorraussetzt, was mit MS4W nicht gegeben ist, kann MapCache für diese Arbeit nicht verwendet werden.

GeoWebCache

Der im Zusammenhang mit GeoServer vorgesstellte TC GWC ist ebenfalls mit MapServer nutzbar. Dessen Fähigkeiten und Konfiguration sind unter 3.3.1[55] beschrieben Es ist möglich, den bestehen GeoServer zu belassen und einzig im GWC einen weiteren Layer in „geowebcache.xml“ hinzuzufügen:

```
<wmsLayer>
        <name>CGIFeldgrenzen</name>
        <mimeFormats>
        <string>image/jpeg</string>
        <string>image/png</string>
        </mimeFormats>
        <parameterFilters>
        <stringParameterFilter>
                <key>map</key>
                <defaultValue>WMSmap.map</defaultValue>
                <values>
                  <string>WMSmap.map</string>
                </values>
        </stringParameterFilter>
```

[53] [WIKI-YAML]
[54] vgl. [MP]
[55] S. 26

```
        <regexParameterFilter>
          <key>shash</key>
          <defaultValue></defaultValue>
          <regex>[\w\d\s\/\+\=\:\%]{20,100}</regex>
        </regexParameterFilter>
        </parameterFilters>
        <wmsUrl>
        <string>http://localhost:84/fcgi-bin/mapserv.exe</string>
        </wmsUrl>
        <wmsLayers>Feldgrenzen</wmsLayers>
        <transparent>true</transparent>
</wmsLayer>
```

Listing 4.5: MapServer Layer Eintrag im GWC

Da GeoWebCache EPSG:3857 nicht interpretieren kann, muss EPSG:900913 genutzt werden.

Metatiling ist nur mit einem TC möglich. MapServer unterstützt diese Funktion nicht, anders als GeoServer. Zwar kann im Mapfile mit den Parametern „tile_map_edge_buffer“ der auszuweitende Rand für die Labeldarstellung gesetzt und mit „tile_metatile_level“ ein Metatile bei jeder Anfrage berechnet werden, aber ist dies nicht so komfortabel wie das Metatiling des GeoServers.
Die Metatiles des MapServers verweilen nicht im Speicher, sodass sie nur einmal verwendet werden können. Damit ist es zwar möglich, dass die Probleme mit doppelten oder verschobenen Labels behoben werden, jedoch hat dies eine starke Leistungsminderung zur Folge.

4.5 Leistung

4.5.1 Empfehlungen

Im Nachfolgenden werden Empfehlungen zur Nutzung von MapServer als Fast-CGI-Skript gegeben.
Wird das MapServer MapScript genutzt, sind die Hinweise der jeweiligen Programmiersprache und allgemeine Programmierhinweise für eine Optimierung des gesamten Vorganges zu berücksichtigen.

Eine Kompilierung der benötigten Bibliotheken und Skripte auf dem zu nutzenden Server bietet sich an, da dadurch keine überschüssigen Services aktiv sind und die Leistung bei selbst kompilierten Systemen erfahrungsgemäß höher ausfällt.

Wie in 4.2.1[56] bereits erwähnt, sollte Fast-CGI genutzt werden, da sich dadurch die Verarbeitungsdauer halbiert.

[56] S. 43

Auch dieser Mapserver kann in mehreren Instanzen betrieben werden. Mit dem Modul „mod_balance“, können Anfragen auf die Instanzen verteilt werden.

Die nachfolgenden Hinweise stammen von [MS-O].

Datenbank
Wie bei GeoServer, wird ebenso eine Indizierung der zu verwendenden Tabellen empfohlen.
„ALTER TABLE table ADD PRIMARY KEY (gid);
CREATE INDEX table_the_geom ON table (the_geom) USING GIST;“
Wäre der auszuführende Befehl.
Mit dem Eintrag „PROCESSING „CLOSE_CONNECTION=DEFER““ in Layern, welche dieselbe Datenquelle nutzen, wird die Datenbankverbindung bis zum Ende der Abarbeitung des Mapfiles aufrechterhalten. Ansonsten wird nach jedem Layer die Verbindung geschlossen. Dadurch wird durch Verhinderung des Unterbrechens und Neuaufbaus der Verbindung Zeit gespart.

Mapfile
Die Projektionen sollten in ausgeschriebener Form (inline projection) und nicht als EPSG Code angegeben werden. Wird der EPSG Code verwendet, muss zunächst dessen Definition aus einer Datei gelesen werden.
Zusätzlich sollte jedes Mapfile nur die notwendigsten Layer beinhalten. Bei vielen Layern sollte man diese gruppieren und in andere Mapfiles auslagern. Neben Zoomstufenabhängiger Darstellung (durch SLD-Datei oder CLASS-Objekt) ermöglicht die Festsetzung erlaubter Zoomstufen (MAX- und MINSCALEDENOM) einzig jene Daten anzuzeigen, welche benötigt werden. Die Anzahl der zu rendernden Daten ist direkt proportional zum Renderaufwand bzw. -dauer.
Es sollten weiterhin nur Rastersymbole und Schriftarten eingebunden werden, welche benötigt werden.

Caching
Die unter der Überschrift Caching in 3.4.1[57] aufgeführten Hinweise gelten hier ebenso. Da MapServer aber keinen Tile Cache per sé besitzt, ist es notwendig einen der unter 4.4[58] vorgestellten zu installieren und konfigurieren.

Bildformat
Der Speicherplatzbedarf und die Qualität der Bilder spielt eine wichtige Rolle. Als Bildtyp können vorgegebene verwendet werden und eigene als „OUTPUTFORMAT“ im Mapfile erstellt werden.
„image/jpeg“ ist wie folgt definiert:

```
OUTPUTFORMAT
        NAME "jpeg"
```

[57] S. 32
[58] S. 51

```
        DRIVER AGG/JPEG
        MIMETYPE "image/jpeg"
        IMAGEMODE RGB
        EXTENSION "jpg"
        FORMATOPTION "GAMMA=0.75"
END
```

Listing 4.6: Mapfile: JPEG

Da die Bildgröße eine wichtige Rolle spielt, kann mit einem eigenen Bildformat die Qualität so niedrig wie nötig gehalten werden:

```
OUTPUTFORMAT
        NAME 'AGG_Q'
        DRIVER AGG/PNG
        IMAGEMODE RGB
        MIMETYPE "image/png"
        FORMATOPTION "QUANTIZE_FORCE=ON"
        FORMATOPTION "QUANTIZE_DITHER=OFF"
        FORMATOPTION "QUANTIZE_COLORS=256"
END
```

Listing 4.7: Mapfile: eigenes Bildformat

Mit dieser Konfigurationsmöglichkeit sollte ein Bildformat genutzt werden, welches die Mindestanforderung der Bildqualität erfüllt und gleichzeitig den geringsten Speicherplatzbedarf besitzt. In MS4W sind nicht alle Bibliotheken vorhanden, um alle unter `http://mapserver.org/output/ogr_output.html` und `http://mapserver.org/output/agg.html` aufgeführten Formate nutzen zu können. So sind nur AGG Formate nutzbar.

4.5.2 Tests

Zum direkten Vergleich der Leistungsfähigkeit, wurden drei Tests entsprechend den unter 2.4[59] beschriebenen Bedingungen durchgeführt und dokumentiert.
Dabei wird direkt, bzw. indirekt bei Nutzung des Tile Caches, das Fast-CGI Skript angesprochen und kein PHP Skript, da dies die Leistungsfähigkeit des MapServer deutlicher wiederspiegelt.

Der Anteil der Datenbank an der Antwortdauer ist sehr gering. Pro Anfrage an den MapServer, wird ein SQL Statement abgefragt, welches die gesamten Daten der WMS getMap Anfrage liefert:
`selectencode(ST_AsBinary(ST_Force_2D("geom"),'NDR'),'hex')asgeom ,"id"fromogeo.feldgrenzenwheregeom&&ST_GeomFromText( 'POLYGON((...))',4326)and(hash='EeGP98dtbGVfRnMugaaadG29ZCw=')`
Außerdem unregelmäßig „SELECT postgis_version()“.
Dazu werden etwa 1 bis 30 Millisekunden benötigt.

[59] S. 12

Der erste Test nutzte keinen Tile Cache und kein Metatiling.

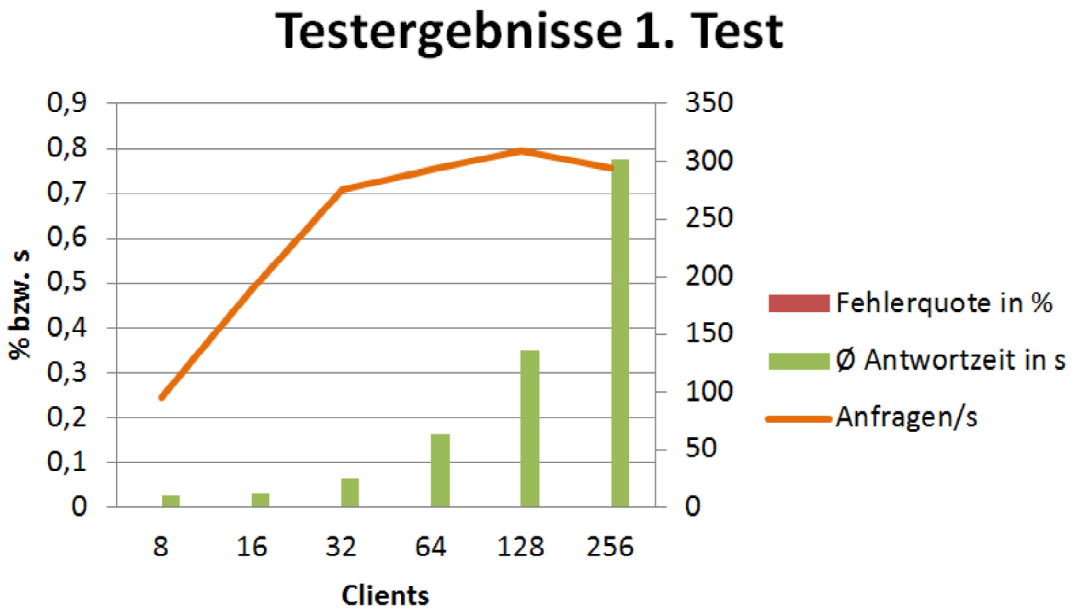

Abbildung 4.3: MapServer 1. Test

Die Leistungsfähigkeit des MapServers wird durch diesen Test deutlich. Selbst bei 256 Clients liegt die Antwortzeit unter einer Sekunde. Die Systemauslastung ist dabei vor Allem durch den MapServer sehr hoch:

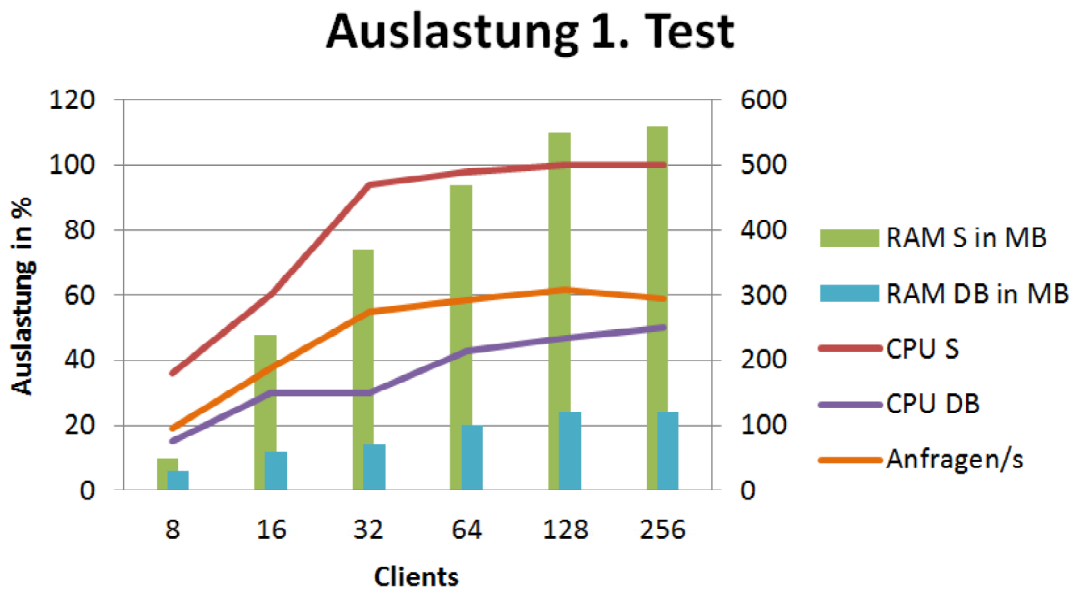

Abbildung 4.4: MapServer Auslastung 1. Test

Für den zweiten Test wird GeoWebCache als Tile Cache eingesetzt. Da es sich um den selben GWC wie im dritten Test in 3.4.2[60] handelt, sind die Ergebnisse dieselben. Einzig die erste Erzeugung der Tiles ist schneller, da MapServer diese rendert.

[60] S. 36

5 Vergleich

Der Zweck dieses Vergleiches ist es, die Höhe der Eignung der beiden MapServer unter gegebenen Anforderungen darzulegen.
Nachdem die Mapserver GeoServer und MapServer analysiert wurden, werden sie nach für den Prototypen relvanten Punkten verglichen.
In 1.2[61] wurden die Analysepunkte Einstellungsmöglichkeiten für dynamische Karten und Leistungsfähigkeit benannt.
Darunter ist konkret zu verstehen:

- Nutzung von dynamischen Layern, welche in der Anfrage Paramter zur Filterung erhalten
- ausreichend sinnvolle Konfigurationsmöglichkeiten des Kartendienstes
- viele Darstellungsmöglichkeiten der Layer
- Nutzung des Entwurfes für zukünftiges AgriPort
- Antwortzeiten von unter einer Sekunde bei größerer Last
- Antwortzeiten von unter 0,5 Sekunden bei geringer Auslastung

Gleichzeitig muss WMS nutzbar und als Datenquelle PostGIS verwendbar sein. Diese zwei wesentlichen Bedingungen sind durch den Prototypen begründet.

Einstellungsmöglichkeiten
Die wichtigste Fähigkeit die ein Mapserver besitzen muss, ist die Unterstützung von WMS. Beide Mapserver erlauben eine OGC-konforme Nutzung dieses Dienstes.

Da GeoServer eine GUI für die Konfiguration bereitstellt, ist die Einrichtung der Datenquellen, der Layer und des WMS Dienstes mit geringem Aufwand möglich.
MapServer muss hingegen in Konfigurationsdateien eingerichtet werden.
Allgemein ist die Handhabung des GeoServers einfacher und verständlicher.

Beide Mapserver ermöglichen eine individuelle Konfiguration der Datenquellen und Layer. Diese ist bei MapServer durch dessen Mapfiles sehr tiefgreifend, sodass in dieser Arbeit nur die wichtigsten Elemente benannt werden konnten.
Durch MapScript ist es möglich, die Funktionen des MapServers zusätzlich in häufig vorkommenden Programmiersprachen zu verwenden.
GeoServer bietet ähnliche Einstellungsmöglichkeiten, besitzt aber keine weiteren Zugänge zur Nutzung der Funktionalitäten. Weiterhin ist es mit diesem nicht möglich, das Bildformat der Karten den eigenen Wünschen anzupassen oder neben SLD andere Gestaltungsmaßnahmen zu ergreifen als mit MapServer.
Dafür bietet GeoServer Labeloptimierungen durch erweiterte SLD Funktionen an.

[61] S. 3

Für die Nutzung des Prototypen eignen sich in diesem Punkt beide Mapserver.
Sollten Label in Zukunft eine Rolle spielen, wären die Möglichkeiten des GeoServers von Vorteil.

Ebenso ist es mit beiden Mapservern möglich dynamische Karten zu verwenden. Dynamik meint eine parametrisierte Abfrage der Daten, sodass jeder Client unterschiedliche Daten erhält.
In beiden Systemen können parametrisierte SQL Abfragen zur Datenbeschaffung genutzt werden.
In diesem Punkt gibt es keine Unterschiede.

Die Datenbeschaffung ist ebenfalls in beiden Systemen für PostGIS einheitlich und komfortabel.

Weitere Einstellungsmöglichkeiten sind die Zusammenfassung von Layern in Gruppenlayer, zoomstufenabhängige Darstellung und die Beschränkung von Daten pro Abfrage.
Beide Mapserver bieten die Möglichkeit Gruppenlayer zu bilden.
MapServer ermöglicht es nicht nur mehrere Layer zusammenzufassen, sondern auch pro Anfrage mehrere Layer abhängig der Einstellungen und Parameter darzustellen.
Mit GeoServer können mehrere Layer oder Gruppenlayer angezeigt werden. Eine datenabhängige Darstellung ist nur mit SLD möglich, was MapServer dagegen unterstützt.
Die zoomstufenabhängige Darstellung wird im Geoserver durch SLD realisiert. MapServer bietet außerdem in der Layerkonfiguration an, abhängig der Zommstufe, festgelegte Elemente zu zeichnen oder außer Acht zu lassen.
Zusätzlich kann in beiden Systemen die Anzahl der auszugebenden Elemente (Features) festgesetzt werden.

Die Größe und Qualität der Bilder ist ebenfalls von Bedeutung.
Um den optischen Anspruch zu wahren, muss die Qualität eines komprimierten Bildes jenes eines unkomprimierten Bildes sehr stark ähneln.
Im Gegensatz dazu muss der Speicherbedarf gering ausfallen, sodass ein qualitativ hochwertiges Bildformat mit geringem Speicherplatzbedarf benötigt wird.
Beide Mapserver ermöglichen die Nutzung von Formaten wie png und jpeg.
Je nach installierter Bibliothek können Stärke der Kompression und weitere Einstellungen zum Bildformat festgesetzt werden. MapServer ermöglich eine genaue Definition des gewünschten Bildformates, wobei durch GeoServer nur die Stärke der Kompression von jpeg und png angegeben werden kann.

In diesem Punkt sind beide Mapserver gleich gut geeignet, da sie die notwendigen Vorrausetzungen erfüllen und verteilte Unterschiede in den Möglichkeiten bestehen.

Einbindung
Eine Einbindung des Mapservers in den bestehenden Entwurf der serverseitigen Logik von AgriPort bietet sich an. Der Aufbau des Entwurfes ist in 2.2[62] beschrieben.
GeoServer kann als Java System zwar plattformunabhängig betrieben werden, müsste aber größtenteils unabhängig der bisherigen Implementierungen genutzt werden.
Einzig die Erstellung eines eigenen Service, welcher Anfragen an den GeoServer weiterreicht, wäre möglich. MapServer dagegen ist bereits Bestandteil des Entwurfes.
Somit bieten sich vier Möglichkeiten der Einbindung, welche unter 4.2.4[63] beschrieben sind.

Für die Nutzung der bereits bestehenden Implementierung ist MapServer geeignet.

Leistung
Während des Praktikums des Autors wurde GeoServer als Mapserver genutzt.
In der Grundkonfiguration war die Antwortzeit bei geringer Belastung sehr hoch, was der Qualität des Prototypen sehr abträglich war. Es wurde deutlich, dass die Leistungsfähigkeit ein wesentliches Kriterium für die Eignung eines Mapservers ist.

Der Prototyp ist für die mobile Nutzung ausgelegt. Dadurch erfolgen die Anfragen aus dem mobilen Internet, wodurch zusätzlich zur Antwortzeit des Mapservers die Übertragungsdauer inerhalb des Netzwerkes hinzukommt.
Durch diese nicht beeinflussbare Verzögerung sollen die genannten Antwortzeiten möglichst unterboten werden.

Aus diesem Grund werden die Tests zu den beiden Mapservern herangezogen und verglichen.

[62] S. 10
[63] S. 47

Unter Last:

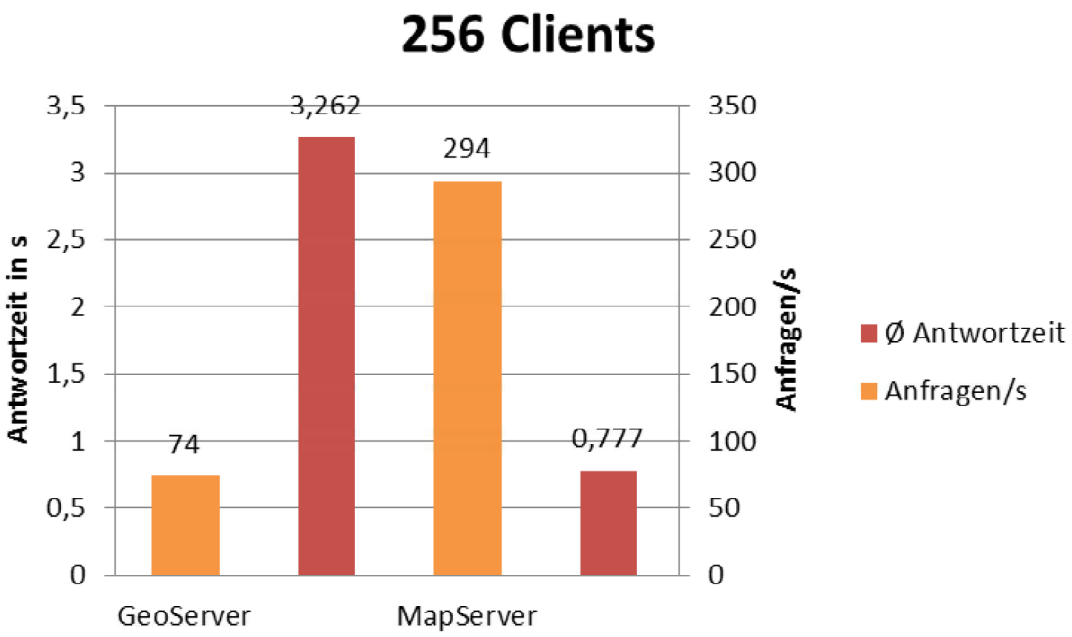

Abbildung 5.1: Vergleich: 256 Clients

Unter geringer Auslastung:

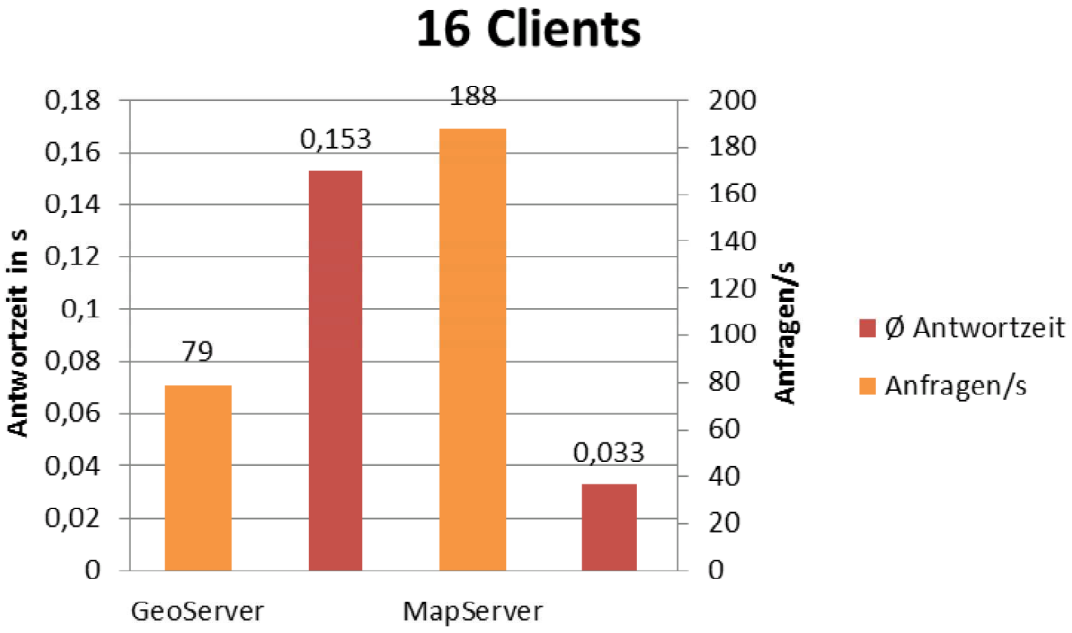

Abbildung 5.2: Vergleich: 16 Clients

Hierbei ist MapServer in beiden Szenarien wesentlich performanter.

MapServer bietet selbst bei Last Spielraum nach oben, sodass die 1-Sekunden-Marke erst bei mehr als 350 Clients erreicht wird. Diese Grenze erreicht GeoServer bereits bei etwa 90 Clients.

Die Fehlerquote war durchgehend bei beiden Mapservern 0 bis 0,2%. Diese Quote ist sehr positiv zu bewerten.

Kommt ein TC für alle Layer zum Einsatz, bestimmt dieser die Leistung des Systems.
Ziel sollte es sein, WMS direkt über den Mapserver sowie über einen TC zu nutzen.
In den Tests wurde nur der Layer der Flurstücke betrachtet. Andere Layer, welche über den Flurstücken liegen, könnten mit einem TC gespeichert werden, sofern es sich um unveränderliche Daten handelt. Dadurch würde eine Entlastung des Mapserver stattfinden.
Jedoch ist MapServer, inklusive einem TC, dem GeoServer mit GWC vorzuziehen, da MapServer die Tiles wesentlich performanter erzeugt. Dadurch ist seeding und reseeding in kürzerer Zeit möglich.

MapServer besitzt in diesen Szenarien eine größere Leistungsfähigkeit als GeoServer, wobei letzterer den Anforderungen an die Leistungsfähigkeit nicht gerecht wird.

Schlussfolgerung
Im Hinblick auf den Vergleich wird MapServer allen Anforderungen gerecht.
Die Empfehlung für die Nutzung eines Mapservers lautet MapServer.
Besonders die vielseitigen Einbindungsmöglichkeiten in den AgriPort Entwurf und die Leistungsfähigkeit machen diesen Mapserver empfehlenswert.

6 Zusammenfassung und Ausblick

Diese Arbeit begann mit der Darlegung der Notwendigkeit der Analyse des Softwareelementes Mapserver.
Daraus wurde die Aufgabe abgeleitet, die relevanten Mapserver entsprechend Analysepunkten zu analysieren und zu vergleichen.

Vor dem Beginn der Analyse musste ein verständlicher Rahmen geschaffen werden.
Dazu wurde das den Mapserver nutzende System vorgestellt und dessen Anforderungen an den Mapserver benannt. Um die Vorarbeiten seitens AgriCon zu berücksichtigen, wurde der serverseitige Entwurf für das Datenportal AgriPort erläutert. Weiterhin wurden relevante Begriffe geklärt und die Testumgebung vorgestellt.

Durch diese Einführung konnte in den Kapiteln drei und vier eine Analyse der beiden relevanten Mapserver GeoServer und MapServer stattfinden. Dabei wurde jeder Mapserver zunächst allgemein vorgestellt. Darauf folgte die Sichtung der Einstellungsmöglichkeiten in Hinsicht auf den Dienst WMS und der speziellen Erweiterungen. Die Erweiterungen unterschieden sich zwischen den Mapservern, wobei das Speichern von Karten mit Hilfe des Elementes Tile Cache bei beiden behandelt wurde.
Die Analyse der beiden Mapserver wurde jeweils durch Lasttests abgeschlossen.

Als Ergebnis konnte im Kapitel fünf ein Vergleich vollzogen werden.
Das sich daraus ergebende Resultat war die Empfehlung des Mapservers MapServer für den Anwendungsfall Agriport Mobile. Die Eignung wurde besonders an Hand der Leistungsfähigkeit und der Möglichkeiten zur Einbindung festgestellt.

Ausblick
In dieser Arbeit wurde nicht nur eine umfassende Darstellung der Mapserver GeoServer und MapServer vollzogen, sondern auch eine wegweisende Empfehlung für Agriport Mobile von AgriCon gegeben.

Somit ist AgriCon nicht nur im Besitzt eines Nachschlagewerkes, vielmehr ist der Beweis erbracht, dass MapServer die beste Lösung für die gesetzten Anforderungen darstellt und für AgriPort und Agriport Mobile gleichermaßen genutzt werden kann.

Es ist somit möglich, den Entwurf weiterzuentwickeln und dabei Agriport Mobile miteinzubeziehen.
Die unter 4.2.4[64] vierte genannte Möglichkeit der Erstellung einer JSON-Schnittstelle analog der SOAP-Schnittstelle soll zukünftig implementiert werden. Damit sind weitere Clients, zusätzlich zu AgriPort und Agriport Mobile, in den Entwurf einbindbar.

[64] S. 47

Für ein Datenportal empfiehlt sich der Einsatz eines Tile Cache für statische Karten. Agriport Mobile wird zukünftig statische Karten verwenden, weshalb ein Tile Cache den Abruf dieser enorm beschleunigen würde.
Wird MapCache in Zukunft den Beta Status verlassen, wäre er der geeignetste, da er performant viele Möglichkeiten bietet und für eine Nutzung mit MapServer konzipiert wurde.
Ebenso ist es möglich, statische Karten von externen Quellen und MapServer mit MapProxy der Version 1.5 im Vorraus zu speichern. Somit wird MapServer entlastet und eine zentrale Instanz für die Verwaltung der gesamten statischen Karten geschaffen.
Siehe dazu 4.4[65].

[65] S. 51

Anhang A: XML-Klassen AgriPort

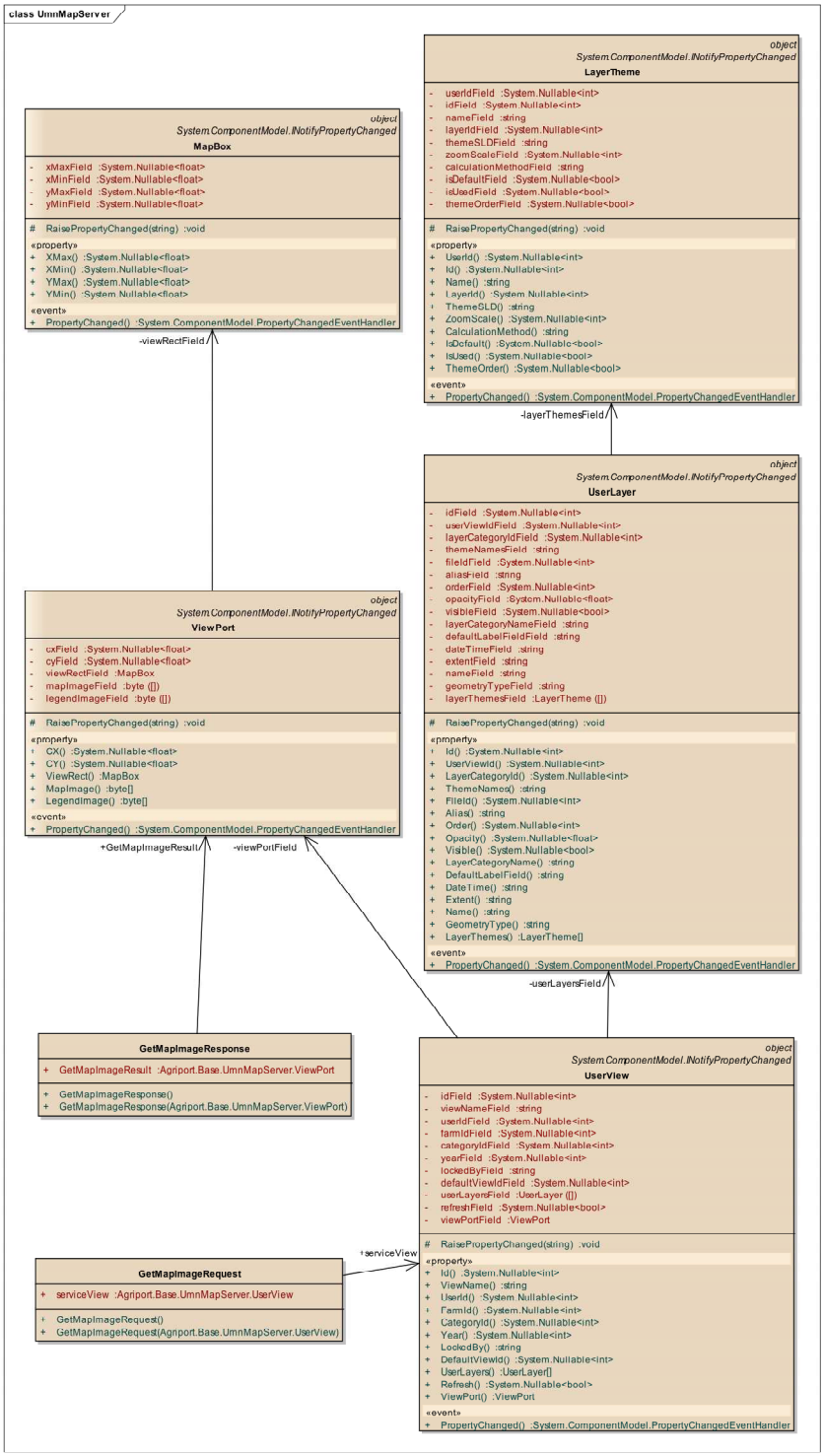

Abbildung A.1: XML-Klassen AgriPort

Anhang B: WMS-konformes Mapfile

```
MAP
        NAME "WMSmap"
        STATUS ON
        SIZE 256 256
        EXTENT 500000 1824380.471488223 5203252.387133993 7400000
        UNITS METERS
        IMAGETYPE 'AGG_Q'

        #DATAPATTERN "[\w\d\s\/\+\=]{50}"

        OUTPUTFORMAT
                NAME 'pngst'
                DRIVER "AGG/PNG"
                MIMETYPE "image/pngst"
                IMAGEMODE RGBA
                EXTENSION "png"
                TRANSPARENT ON
        END

        OUTPUTFORMAT
                NAME 'AGG_Q'
                DRIVER AGG/PNG
                IMAGEMODE RGB
                MIMETYPE "image/png"
                FORMATOPTION "QUANTIZE_FORCE=ON"
                FORMATOPTION "QUANTIZE_DITHER=OFF"
                FORMATOPTION "QUANTIZE_COLORS=256"
        END

        WEB
          IMAGEPATH "C:/ms4w/Apache/htdocs/tmp/"
          IMAGEURL "tmp/"
          METADATA
                WMS_TITLE "WMSmap"
                WMS_ONLINERESOURCE "http://localhost:84/fcgi-bin/mapserv?map=WMSmap
                    .map&"
                WMS_SRS "EPSG:3857"
                WMS_ENABLE_REQUEST "GetMap␣GetFeatureInfo␣GetCapabilities␣
                    GetLegendGraphic"
          END
        END

        PROJECTION      #EPSG:3857
                "proj=merc"
                "a=6378137"
                "b=6378137"
                "lat_ts=0.0"
                "lon_0=0.0"
                "x_0=0.0"
                "y_0=0"
                "k=1.0"
                "units=m"
                "nadgrids=@null"
                "wktext"
                "no_defs"
        END

        LEGEND
                IMAGECOLOR 255 255 255
                OUTLINECOLOR 80 80 80
                KEYSIZE 30 20
```

```
        END

        LAYER
                NAME "Feldgrenzen"
                TYPE POLYGON
                VALIDATION
                  'shash' '[\w\d\s\/\+\=]{28}'
                END
                CLASS
                        NAME "polygon"
                        STYLE
                                OUTLINECOLOR 255 255 255
                                COLOR 170 170 170
                                #BACKGROUNDCOLOR 1 1 1
                                WIDTH 3
                                OPACITY 60
                        END
                END
                CONNECTION "user=dboperator␣password=dbugis␣dbname=agrodb␣host
                    =192.168.0.3␣port=5432"
                CONNECTIONTYPE postgis
                PROCESSING "CLOSE_CONNECTION=DEFER"
                DATA "geom␣from␣ogeo.feldgrenzen␣using␣unique␣id␣using␣srid=4326"
                DUMP false
                EXTENT 4.49157642122284 16.16967511640222 46.741611471444564
                    55.196005052343054 #EPSG:4326
                #EXTENT 500000 1824380.471488223 5203252.387133993 7400000 #EPSG
                    :3857
                FILTER "hash='%shash%'"
                OPACITY 60
                #PROJECTION
                        #"init=epsg:3857"
                #END
                PROJECTION
                        #"init=epsg:4326"
                        "proj=latlong"
                        "ellps=WGS84"
                        "datum=WGS84"
                END

                #SIZEUNITS meters
                STATUS Default
                TOLERANCE 0
                TOLERANCEUNITS pixels
                TRANSFORM true
                #UNITS meters
                METADATA
                        "wms_title" "Feldgrenzen"
                        #"wms_srs" "EPSG:3857"
                        "wms_extent" "4.49157642122284␣16.16967511640222␣
                            46.741611471444564␣55.196005052343054"
                        "wms_enable_request" "GetMap␣GetFeatureInfo␣GetCapabilities
                            ␣GetLegendGraphic"
                END
        END
END
```

Listing B.1: WMS-konformes Mapfile

Anhang C: Aufbau Mapfile

Dieser Abschnitt beschreibt den grundsätzlichen Aufbau eines Mapfiles.
Zusätzlich werden die wichtigesten Elemente aufgeführt.

Ein Mapfile besteht aus dem Objekt „MAP“, welches weitere Objekte und Attribute enthält. Ein jedes Objekt beginnt mit dessen Name und endet mit „END“. Beispiel:

```
MAP
        ...
        LAYER
                ...
        END

        LAYER
                ...
        END
        ...
END
```

Die Attribute des Objektes „MAP“ werden Mapheader genannt. Mit diesem Mapheader ist es möglich, den Namen, den Status, die Größe, die Ausdehnung, den Bildtyp und andere Eigenschaften festzulegen. Beispiel:

```
MAP
        NAME "Map"
        STATUS ON
        SIZE 400 400
        EXTENT 196900 8027100 245000 8073000
        UNITS METERS
        IMAGETYPE jpeg

        ...
END
```

Die Nachfolgenden Objekte werden, falls nicht anders beschrieben, direkt im „MAP“ Objekt angegeben.

Als Bildtyp können vorgegebene verwendet werden und eigene als „OUTPUTFORMAT“ erstellt werden. Der im oberen Beispiel verwendete Bildtyp „jpeg“ ist wie folgt definiert:

```
OUTPUTFORMAT
  NAME "jpeg"
  DRIVER AGG/JPEG
  MIMETYPE "image/jpeg"
  IMAGEMODE RGB
  EXTENSION "jpg"
  FORMATOPTION "GAMMA=0.75"
END
```

Da die Bildgröße eine wichtige Rolle spielt, kann mit einem eigenen Bildformat die Qualität so niedrig wie nötig gehalten werden:

```
OUTPUTFORMAT
        NAME 'AGG_Q'
        DRIVER AGG/PNG
        IMAGEMODE RGB
        MIMETYPE "image/png"
        FORMATOPTION "QUANTIZE_FORCE=ON"
        FORMATOPTION "QUANTIZE_DITHER=OFF"
        FORMATOPTION "QUANTIZE_COLORS=256"
END
```

Das Objekt „WEB“ bestimmt zusätzliche Rückgabeelemente, den Speicherort der Bilder, den Zoombereich und Metainformationen zum Dienst. Beispiel:

```
WEB
  FOOTER "footer.htm"
  IMAGEPATH "/ms4w/tmp/ms_tmp/"
  IMAGEURL "/ms_tmp/"
  MAXSCALEDENOM 24000
  MAXSCALEDENOM 100
  METADATA
        title "Test"
        author "K. Junghanns"
  END
END
```

„REFERENCE“ ermöglicht die Einbindung eines Bildes oder Symbols in die Karte.

```
REFERENCE
        IMAGE /ms4w/tmp/ms_tmp/efate.png
        EXTENT 196900 8027100 245000 8073000
        STATUS ON
        OUTLINECOLOR 255 0 0
        SIZE 50 50
END
```

Um verschiedene Koordinatenreferenzsysteme verwenden zu können oder eine Umprojektion durchzuführen, muss mit „PROJECTION“ das Ausgabe-Referenzsystem angegeben oder beschrieben werden.

```
PROJECTION
        "init=epsg:4326"
END
```

bzw.

```
PROJECTION
  "proj=latlong"
  "ellps=WGS84"
  "datum=WGS84"
END
```

beziehen sich auf EPSG:4326.

Die Layer werden durch das Objekt „LAYER“ repräsentiert. Dieses besitzt nach [MS-MF] eine Vielzahl von Attributen:

```
LAYER
        CLASS
                ...
        END
        CONNECTION "user=nobody password=****** dbname=dbname host=localhost port
            =5432"
        CONNECTIONTYPE postgis
        DATA "geom from ogeo.feldgrenzen using unique id using srid=4326"
        DUMP false
        EXTENT 4.49157642122284 16.16967511640222 46.741611471444564
            55.196005052343054
        FEATURE
                ...
        END
        FILTER "geom!=NULL"
        GRID
                ...
        END
        JOIN
                ...
        END
        LABELCACHE on
        LABELMAXSCALEDENOM 24000
        LABELMINSCALEDENOM 100
        LABELREQUIRES "[summenkarte]"
        MAXFEATURES 1000
        MAXSCALEDENOM 24000
        METADATA
                ...                             #Metainformationen
        END
        MINSCALEDENOM 100
        NAME "Feldgrenzen"
        OPACITY 60
        POSTLABELCACHE false
        PROCESSING "CLOSE_CONNECTION=DEFER"
        PROJECTION
                "init=epsg:4326"
        END
        REQUIRES "[summenkarte]"
        SIZEUNITS meters
        STATUS on
        SYMBOLSCALEDENOM 1000
        Template
                ...                             #Ausgabe in einer Template-HTML Datei
        END
        TILEINDEX
                ...
        END
        TOLERANCE 0
        TOLERANCEUNITS pixels
        TRANSFORM true
        TYPE polygon
        UNITS meters
END
```

Die Auswirkungen eines jeden Attributes könnnen unter [MS-MF] nachgelesen werden.

Jedes „LAYER“ Objekt kann „CLASS“ Objekte enthalten, welche die Darstellung als Karte, ähnlich SLD, festlegen. Beispiel:

```
CLASS
        NAME "polygon"
        STYLE
                OUTLINECOLOR 255 255 255
                COLOR 170 170 170
                WIDTH 1
                OPACITY 60
        END
END
```

Die Definition der Legende findet direkt im „MAP“ Objekt statt:

```
LEGEND
        IMAGECOLOR 255 255 255
        OUTLINECOLOR 80 80 80
        KEYSIZE 30 20
END
```

Literaturverzeichnis

[1h-mb] 1h o.V.: „Apache Module mod_balance“, o.J., http://www.1h.com/opensource/mod_balance.html, Abruf: 23.09.2012

[AB12] agrar.bayer.de o.V.: „DLG-Feldtage 2012 der erwartete Besuchermagnet“, 2012, http://agrar.bayer.de/DLG_Feldtage_2012_der_erwartete_Besuchermagnet.cms, Abruf: 2.07.2012

[FÜ01] Fürpaß, Christian: „Mapserver als Hilfsmittel zur Datenvisualisierung im Internet“, Diplomarbeit, Universität Wien, 2001

[GeoS11] GeoSolution: „GeoServer on steroids“, 2011, Präsentationsfolien, FOSS4G 2001 in Denver

[GS-JAVA] GeoServer.org: „Java Considerations“, 2012, http://docs.geoserver.org/2.1.x/en/user/production/java.html, Abruf: 23.09.2012

[GS-PG] GeoServer.org: „PostGIS“, 2011, http://docs.geoserver.org/2.1.3/user/data/postgis.html, Abruf: 06.08.2012

[GS-SLD] GeoServer.org: „Parameter substitution in SLD“, 2011, http://docs.geoserver.org/2.1.3/user/styling/sld-extensions/substitution.html, Abruf: 11.08.2012

[GS-VP] GeoServer.org: „WMS vendor parameters“, 2011, http://docs.geoserver.org/2.1.3/user/services/wms/vendor.html, Abruf: 10.08.2012

[GWC-REST] GeoWebCache.org: „REST API“, 2012, http://geowebcache.org/docs/current/rest/index.html, Abruf: 23.09.2012

[GWIKI-WFS] GISWIKI o.V.: „Web Feature Service“, http://en.giswiki.net/wiki/Web_Feature_Service, Abruf: 07.08.2012

[ITW-MG] ITWissen o.V.: „Mooresches Gesetz“, o.J., http://www.itwissen.info/definition/lexikon/Mooresches-Gesetz-Moores-law.html, Abruf: 22.09.2012

[JU10] Jurk, Thomas: „Integration von Tiled Map Services in Geodateninfrastrukturen“, Diplomarbeit, Hochschule für Technik und Wirtschaft Dresden, 2010

[JU12] Junghanns, Kurt: „Praktikumsbericht“, Praktikumsbericht, Hochschule Mittweida, 2012

[MP] MapProxy: „Sources“, 2012, http://mapproxy.org/docs/nightly/sources.html, Abruf: 12.09.2012

[MS-A] MapServer: „About“, 2011, http://mapserver.org/about.html#about, Abruf: 14.08.2012

[MS-L] MapServer: „Label“, 2011, http://mapserver.org/mapfile/label.html, Abruf: 24.09.2012

[MS-MF] MapServer: „Mapfile“, 2011, http://mapserver.org/de/mapfile/index.html, Abruf: 01.09.2012

[MS-O] MapServer: „Optimization“, 2011, http://mapserver.org/de/optimization/index.html, Abruf: 16.09.2012

[MS-W] MapServer: „A Simple CGI Wrapper Script“, 2011, http://mapserver.org/cgi/wrapper.html, Abruf: 24.09.2012

[MS-WMS] MapServer: „WMS Server“, 2011, http://mapserver.org/ogc/wms_server.html#setting-up-a-wms-server-using-mapserver, Abruf: 24.09.2012

[NE09] Neubert, Matthias: „Konzeption und Umsetzung eines mobilen, kartenbasierten Informationssystems“, Diplomarbeit, Technische Universität Dresden, 2009

[O-FAQ] Oracle: „Frequently Asked Questions About the Java HotSpot VM“, o.J., http://www.oracle.com/technetwork/java/hotspotfaq-138619.html#64bit_selection, Abruf: 09.08.2012

[OG10] OpenGeo: „GeoServer in Production“, 2010, http://opengeo.org/publications/geoserver-production/, Abruf: 19.07.2012

[OGC-WFS] Vretanos, Panagiotis A.: „Web Feature Service Implementation Specification“, OGC OpenGIS Implementation Specification, 2005

[O-JP] Oracle: „Java HotSpot VM Options“, o.J., http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html, Abruf: 06.08.2012

[OSGeo-GS] OSGeo o.V.: „GeoServer“, o.J., http://live.osgeo.org/de/overview/geoserver_overview.html, Abruf: 22.09.2012

[OSGeo-OGC] OSGeo o.V.: „Open Geospatial Consortium“, o.J., http://wiki.osgeo.org/wiki/Open_Geospatial_Consortium, Abruf: 22.09.2012

[OSGeo-PG] OSGeo Mailing List: „PostGIS datasource and prepared-Statements“, 2012, http://osgeo-org.1560.n6.nabble.com/PostGIS-datasource-and-preparedStatements-td4578274.html, Abruf: 07.08.2012

[OSGeo-WMSC] OSGeo o.V.: „WMS Tile Caching“, o.J., http://wiki.osgeo.org/wiki/WMS_Tile_Caching, Abruf: 25.09.2012

[PR06] Prüller, Rainer: „Visualisierung von Geodaten mit dem UMN Mapserver“, Magisterarbeit, Institut für Geoinformation der Technischen Universität Graz, 2006

[RÖ10] Röseberg, Franziska: „Precision Farming kommt“, 2008, http://www.innovations-report.de/html/berichte/agrar_forstwissenschaften/bericht-104352.html , Abruf: 2.07.2012

[SC07] Schütze, Emanuel: „Stand der Technik und Potenziale vonSmart Map Browsing im Webbrowser“, Diplomarbeit, Hochschule Bremen, 2007

[TC] TileCache: „TileCache - Web Map Tile Caching“, 2010, http://tilecache.org/, Abruf: 27.09.2012

[WIKI-JAI] Wikipedia o.V.: „Java Advanced Imaging“, o.J., http://de.wikipedia.org/wiki/Java_Advanced_Imaging, Abruf: 09.08.2012

[WIKI-KML] Wikipedia o.V.: „Keyhole Markup Language“, o.J., http://de.wikipedia.org/wiki/Keyhole_Markup_Language, Abruf: 08.08.2012

[WIKI-MS] Wikipedia o.V.: „MapServer“, o.J., http://de.wikipedia.org/wiki/MapServer, Abruf: 14.08.2012

[WIKI-OGC] Wikipedia o.V.: „Open Geospatial Consortium“, o.J., http://de.wikipedia.org/wiki/Open_Geospatial_Consortium, Abruf: 21.09.2012

[WIKI-WMTS] Wikipedia o.V.: „WMTS“, o.J., http://de.wikipedia.org/wiki/Web_Map_Tile_Service, Abruf: 21.09.2012

[WIKI-YAML] Wikipedia o.V.: „YAML“, o.J., http://de.wikipedia.org/wiki/YAML, Abruf: 06.09.2012

[WIKI-Bench] Wikipedia o.V.: „Benchmark“, o.J., http://de.wikipedia.org/wiki/Benchmark_(EDV)#Software-Benchmarks, Abruf: 09.09.2012

Listingverzeichnis

Glossar

AgriPort Datenportal der Firma AgriCon GmbH, welches eine automatische Verarbeitung und Darstellung der Precision Farming - Daten ermöglicht.

ArcSDE Das Unternehmen ESRI bietet viele Lösungen im GIS Bereich an. Eine ist ArSDe, was eine DBMS Erweiterung für räumliche Datenverwaltung darstellt.

DB2 Kommerzielles relationales DBMS von IBM.

EPSG-Code Weltweit eindeutige Schlüsselnummer für Koordinatenreferenzsystem, mit welchem eine kurze Definition des Koordinatenreferenzsystems möglich ist.

Garbage Collector Automatische Speicherbereinigung, zum Erhalt der Stabilität und Veringerung des Speicherbedarfes.

GDAL Geospatial Data Abstraction Library ist eine Programmbibliothek zur Übersetzung von Raster- und Vektordaten.

GeoTiff TIFF Bild mit verlustfreier Speicherung und zusätzlichen Informationen zum Koordinatenreferenzsystem un den Koordinaten.

GeoTools OpenSource Java GIS Toolkit, welches alle aktuellen Standards des OGC verwendet.

HTTP Hypertext Transfer Protocol ist ein statusloses Netzwerkprotokoll, wobei die Kommunikation mit Requests und Responses erfolgt.

JDBC Java Database Connectivity ist eine einheitliche Datenbankschnittstelle zur Kommunikation zwischen Javaprogramm und Datenbank.

JMeter In Java geschriebenes Last- und Performancetestprogramm.

Lasttest Ist ein Test einer Software, in welchem extreme Last auf das zu testende System einwirkt. Parallel zur Belastung findet eine Beobachtung des Verhaltens statt, sodass Fehler entdeckt oder Verbesserungsmöglichkeiten offen gelegt werden können.

Layer Kategorisierter Teil des Datenbestandes, welcher verwaltet und angezeigt werden kann.

LocalStorage Clientseitiger Schlüssel-Wert Speicher, welcher mit HTML5 wingeführt wurde.

Manifold System Kostenpflichtiges Geoinformationssystem der Firma manifold.net für das Betriebssystem Windows.

MapScript Bibliothek zur Bereitstellung der MapServer-Funktionen in den Programmiersprachen C, C++, PHP, Python, Ruby, Java, Perl und C#/.NET.

MapServer MapServer ist ein freier Mapserver der Open Source Geospatial Foundation (OSGeo), findet eine breite Verwendung und gilt als stabil und Leistungsstark.

Mapserver Internet Map Server, welcher geographische Daten in Form von Bildern und Objekten bereitstellt.

MS4W MapServer for Windows ist ein vorgefertigtes Paket für das Betriebssystem Windows, welches eine grundlegende Nutzung des MapServers ermöglicht.

Objektrelationaler Mapper Dient der Abbildung von Objekten in relationalen Systemen, sodass diese Objekte relational abgespeichert und aus dem relationalen System als Objekte ausgelesen werden können.

OGC Open Geospatial Consortium, eine Organisation zur Aufarbeitung von Geodaten.

OpenSource Bezeichnung für Software, welche unter einer quelloffenen Lizenz steht, wodurch der Quelltext frei zugänglich und veränderbar ist sowie verbreitet werden darf.

PostGIS PostGIS erweitert die objekt-relationale Datenbank Postgre um geographische Objekte und Funktionen, wodurch die daraus entstehende Geodatenbank für GIS Systeme nutzbar wird.

Precision Farming Dieser Präzisionsackerbau zielt auf eine individuelle Betrachtung und Bewirtschaftung einzelner Flurstücke ab, wodurch Unterschiede des Bodens und die variierende Ertragsfähigkeit innerhalb eines Feldes berücksichtigt werden.

reseeding Erneutes seeding von vorhandenen Kartenausschnitten, um das Kartenmaterial aktuell zu halten.

seeding Erzeugung und Speicherung von Kartenausschnitten in Form von Tiles vor deren Abruf durch Clients.

SLD Styled Layer Descriptor ist ein XML-Schema, welches das Aussehen des zu rendernden Layers festlegt.

SOAP Simple Object Access Protocol ist ein verbindungsorientiertes Netzwerkprotokoll, in welchem der Datenaustausch über XML Dateien stattfindet.

Tile Bildausschnitt einer Karte, welcher durch einen Tile-Cache gespeichert wird.

truncating Löschung von Tiles, welche vorrangig durch seeding erzeugt wurden.

WCS Der OGC Standard Web Coverage Service erlaubt die selektierte Abfrage von Rasterdaten mit zusätzlichen Informationen.

WMS Web Map Service ermöglicht als Webservice den Zugriff auf aus Vektor- und Rasterdaten generierten Karten und dazugehörige Informationen.

WMTS Web Map Tile Service „definiert einen Webservice, um digitale Karten kachelbasiert anbieten und abrufen zu können“[66].

YAML Vereinfachte Auszeichnungssprache zur Datenserialisierung.

[66] [WIKI-WMTS]

Zend PHP Framework, welches zusätzlich und unter anderem Web-Technologien, -Services, Suchfunktionen, PDF-Erstellung, Möglichkeiten des Datenbankzugriffes zur Verfügung stellt.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 5.10.2012